Universitat de Lleida
Escola Politècnica Superior
Grau en Enginyeria Informàtica

TREBALL DE FI DE GRAU

Desenvolupament d'un Videojoc amb Unity

Autor: Felip Nicuesa Guilera
Director: Jordi Planes Cid

Juny de 2016

Felip Nicuesa Guilera

Agraïments

Als meus pares i germà per tot el que han fet per mi durant aquests anys.

Al meu tutor, Jordi, per tota l'ajuda proporcionada.

Als meus amics per fer de beta-testers i valorar el joc.

Índex

Índex d'Il·lustracions

Introducció

Motivació

Des de ben petit, els jocs han sét una part molt important de la meva vida, ja sigui per la gran quantitat de jocs amb els que he jugat, les hores que he estat jugant-los, els amics que he fet o la satisfacció i tranquil·litat a l'estar jugant.

Sempre m'ha fascinat com amb un simple joc es poden despertar tants sentiments com poden ser alegria, companyonia, relaxació, tristesa o fins i tot cabreig, com un joc atreu tanta gent diferent, d'arreu del món, d'edats diferents, de gèneres diferents, ètnies o religions diferents en un mateix equip i no hi ha cap tipus de diferència. Poques coses a la vida han agrupat persones tant diferents sota una mateixa cosa, i els videojocs ho han aconseguit.

Tenir l'oportunitat de realitzar un videojoc és com el somni de tota una vida jugant a videojocs. Poder fer, tot i que a una escala molt més petita, que la gent es diverteixi i s'oblidi dels seus problemes durant uns minuts és l'objectiu principal que haurien de compartir tots els desenvolupadors que fan videojocs, sigui per la plataforma que sigui i del gènere que sigui.

També, intento descobrir el dur i les hores de treball que es necessiten pel desenvolupament d'un videojoc. Sé que no és el mateix que fer un videojoc gran, com és un Triple A, o multijugador, ja que es necessiten artistes, dissenyadors, publicistes, etc.. però tot i treballar en un videojoc més petit les ganes i l'esforç seran les mateixes i podré viure de primera mà com és tot el desenvolupament, des de zero, i totes les experiències que això comporta.

Aquests són els meus objectius respecte aquest treball de fi de grau, desenvolupar el meu videojoc des de zero, no solament programar sinó també dissenyar, buscar els requeriments, testing, els problemes que pugin sorgir i moltes d'altre coses. Seguir aprenent de Unity i de desenvolupament de videojoc em motiva per fer aquest treball i l'afronto amb moltes ganes.

L'elecció del tipus de joc tampoc va ser fàcil, tot i que estava una mica limitat pels meus coneixements i per les hores. Vaig valorar fer un joc en 3D, però hi ha pocs models gratis a internet i els que hi ha són poc complerts. Per fer un joc amb 3D necessites de més personal, ja sigui més programadors, un dissenyador, un responsable de projecte, també necessites artistes que et puguin fer els models, les animacions, el decorat, etc. Un cop descartat el 3D, en vaig centrar en tipus de jocs en 2D. Tenia en ment 3 tipus, un Tower Defense, un de Plataformes o algun tipus de joc Educatiu. Després d'un temps de pensar i mirar la quantitat d'assets gratuïts que hi havia per els tipus de jocs, em vaig decantar per un joc de plataformes ja que sóc un gran fan de Mario i sempre m'ha agradat. Aquest factor va ser un factor clau en la meva decisió.

Planificació

Per tal de planificar-me el projecte, treballaré amb la metodologia SCRUM. Una metodologia àgil que em permetrà centrar-me en un aspecte concret del joc durant cada sprint.

Cada sprint tindrà una durada d'entre 1 i 2 setmanes, la durada específica de cada sprint s'ajustarà depenent de la facilitat que tingui fent les tasques assignades, però no es podrà allargar més d'aquest període determinat anteriorment. El projecte tindrà un total de 9 sprints distribuïdes de la següent manera:

- Anàlisi
  En aquest primer sprint, estaré treballant en els requeriments, tant funcionals com no funcionals, de l'aplicació i els casos d'ús.
- Implementar moviments bàsics jugadors. Mockups interfície del joc.
  En aquest punt ja començaré el desenvolupament del videojoc implementant els moviments bàsics dels jugadors, com saltar, caminar i atacar. En aquesta tasca també buscaré els assets corresponents als jugadors. Per últim faré uns mockups de la interfície del joc, tant in-game com dels menús.
- Implementar moviments bàsics enemics. Funcionalitats específiques de cada jugador.
  Seguint el sprint anterior, buscaré els assets pels enemics i començaré la implementació dels seus moviments bàsics que dependran del tipus d'enemics. Ja pot ser volar, caminar, atacar o saltar. L'altra part important del sprint és pensar i implementar les habilitats específiques de cada jugador.
- Disseny nivells del joc
  En aquest sprint dissenyaré 1 o 2 nivells per tal que la diversió, la dificultat i la durada estiguin ben ajustats.
- Disseny menús
  Buscar i dissenyar els menús són les tasques d'aquest sprint.
- Implementar els nivells
  Implementar els nivells dissenyats en anteriors iteracions ficant els enemics, les plataformes i els objectes que hi puguin haver distribuïts pel mapa.
- Implementar el flux entre menús
  Igual que en el sprint anterior, implementar els menús dissenyats anteriorment. També inclou els fons de cada menú i alguna música de fons.
- Funcionalitats pròpies del joc
  Aquest sprint és per acabar de polir el joc, que tot funcioni bé i que no doni cap error a l'editor.
- Testeig
  Fer el testeig del joc segons una plantilla, interpretar els resultats i corregir els possibles problemes.

Cal destacar que com al final de cada sprint ha d'haver un producte funcional, ja es realitzarà un testing per tal que les tasques realitzades en aquell sprint funcionin correctament i no hagi cap incompatibilitat amb la funcionalitat desenvolupada en els sprints anteriors.

Pel que fa a l'administració de les tasques del projecte, treballaré amb una aplicació web anomenada Asana, que et permet crear, modificar, eliminar tasques, assignar-les a la persona que les hagi de dur a terme i marcar cada tasca individualment com a completada. Així d'una manera fàcil i organitzada pots administrar les tasques del teu projecte.

Pressupost

El pressupost és un dels aspectes més importants a tenir en compte a l'hora de desenvolupar un videojoc, ja que pot fer que un joc és desenvolupi o no i també pot fer que es deixi a mig desenvolupament.

Per tant, intentaré minimitzar els costos per tal d'obtenir el màxim benefici, suposadament.

Cost de personal

Lloc de Treball	Número d'hores	Cost per hora	Total (€)
Analista	30	30 €	900
Programador	200	25 €	5000
			5900

Cost de software

Pel desenvolupament del treball es farà servir tant programes de codi obert com programes de software propietari. La utilització d'aquests programes propietaris és a causa que ofereixen millors prestacions que els seus respectius de codi obert. Tot i això, per fer córrer Unity és necessita Windows o Mac OS X i els dos són software propietari.

Producte	Total (€)
Windows 10 Home	135
Office 2016	149
Unity Personal Edition	0
Bitbucket Account	0
SourceTree	0
Asana Account	0
Visual Studio Community	0
Gimp 2.0	0
	284

Cost de hardware

En aquest apartat es contarà el valor del ordenador utilitzat, tot i que ja disposava d'aquest ordenador, inclouré el cost que suposaria adquirir-lo. També es contarà el cost d'energia requerida per fer funcionar aquest ordenador. El càlcul del cost del ordenador està basat en una estimació de vida útil de 5 anys.

Màquina	Cost (€)	Hores dedicació	Total (€)
Ordenador portàtil Asus g551jx	900	230	45
Energia (300W)	0.15	230	10,35
			55,35

Cost de l'Art

L'art l'he tingut de treure d'internet ja que jo no tenia els coneixements per dissenyar i realitzar l'art. Per un joc d'aquestes característiques no crec que faci falta contractar un artista. Tot l'art que he trobat ha set de software lliure i gratuïts.

Art	**Cost (€)**
Pack PixelFantasy	0
TileSets	0
Assets d'enemics	0
Musica i/o efectes de sons	0
	0

Resum Costos

Aquí tenim els resum de costos del projecte, amb i sense I.V.A.

Descripció	**Cost (€)**
Personal	5900
Software	284
Hardware	55,35
Art	0
	6239,35

Descripció	**Cost (€)**
Sense I.V.A.	6239,35
I.V.A. 21%	1310,2635
	7549,61

Anàlisi

Anàlisi de Requeriments

A continuació es presenten els requeriments del videojoc, tant funcionals com no funcionals. Per facilitat la comprensió es presentarà una plantilla de com són els requeriments i els seus camps.

Identificació:	
Títol:	
Descripció:	
Prioritat:	
Validació:	

Cada camp significa el següent:

- *Identificació*: Codi que identifica de forma única cada requeriment.
- *Títol*: Nom descriptiu del requeriment.
- *Descripció*: Petita descripció explicativa i concisa del requeriment.
- *Prioritat*: Valoració de 1 a 4 de la importància d'aquell requeriment dins del videojoc.
- *Validació*: Tests per validar si aquell requeriment funciona.

Requeriments Funcionals

Identificació: RF01	
Títol:	Iniciar una nova partida
Descripció:	El sistema haurà de començar una nova partida sempre que l'usuari ho desitgi. En cas que n'hi hagi una de guardada, serà substituïda.
Prioritat:	3
Validació:	- Usuari obre el joc per primera vegada i es carrega una nova partida. - Usuari decideix iniciar una nova partida, es borra la partida guardada i comença una nova partida.

Identificació: RF02	
Títol:	Guardar una partida
Descripció:	El sistema haurà de guardar una partida, substituint la que hi havia amb anterioritat.
Prioritat:	3
Validació:	- L'usuari es passa un nivell, tanca el joc, el torna a obrir i aquell nivell ja esta passat.

Identificació: RF03	
Títol:	Mostar instruccions del joc

Descripció:	El sistema haurà de proporcionar de manera visual i clara les diferents accions que es poden dur a terme i la tecla corresponent a la acció.
Prioritat:	2
Validació:	- L'usuari entra al joc i a l'opció de Instruccions es mostra les accions amb la tecla corresponent.

Identificació: RF04	
Títol:	Sortir de l'aplicació
Descripció:	L'aplicació s'haurà de poder tancar correctament a través d'un menú, a part de tancant per la creueta de la finestra.
Prioritat:	2
Validació:	- L'usuari prem la tecla ESC per sortir l'aplicació i se li mostra un menú per decidir si tancar o no.

Identificació: RF05	
Títol:	Interactuar amb el personatge
Descripció:	El sistema ha de reconèixer les tecles que es premen i moure el personatge amb conseqüència i de manera correcta.
Prioritat:	4
Validació:	- L'usuari pren consciencia de quines tecles serveixen per moure el personatge, entra a un nivell, prem aquelles tecles i el personatge es mou en la direcció/acció correcta.

Identificació: RF06	
Títol:	Reproduir música i/o efectes de so
Descripció:	El sistema reprodueix diferents músiques segons el menú del joc i reprodueix sons al moure el personatge.
Prioritat:	1
Validació:	- L'usuari obra l'aplicació i als diferents menús escolta una música. - L'usuari entra a un nivell i al moure el personatge, sent efectes de so.

Identificació: RF07	
Títol:	Guardar estadístiques/puntuacions
Descripció:	El sistema ha de guardar les puntuacions i/o estadístiques dels diferents nivells, com per exemple la màxima puntuació d'un nivell.
Prioritat:	3
Validació:	- L'usuari es passa un nivell i obté una puntuació X, al següent cop obté X+1 i li reconeix com a una puntuació major.

Identificació: RF08	
Títol:	Seleccionar un nivell
Descripció:	El sistema t'ha de deixar triar entre els nivells disponibles, sempre i quan estiguin desbloquejats.
Prioritat:	4
Validació:	- L'usuari entra per primer cop i li deixa triar sol el nivell 1. - L'usuari és passa el nivell 1, i li deixa triar entre el nivell 1 i 2.

Identificació: RF09	
Títol:	Intel·ligència Artificial Enemics
Descripció:	El sistema ha de fer que els enemics tinguin diferents intel·ligències per tal de matar/perseguir al jugador.
Prioritat:	4
Validació:	- L'usuari entra a un nivell i els enemics es comporten amb una certa intel·ligència, ja sigui perseguint o actuant racionalment.

Identificació:RF10	
Títol:	Menú Principal
Descripció:	El sistema ha de iniciar-se amb una pantalla principal que tindrà les següents opcions: - Iniciar partida - Instruccions - Opcions
Prioritat:	4
Validació:	- L'usuari entra a l'aplicació i se li mostra un menú principal

Identificació: RF11	
Títol:	Menú Selecció Nivell
Descripció:	El sistema ha de carregar un menú per seleccionar un nivell, tant després del menú principal, com al acabar un nivell.
Prioritat:	4
Validació:	- L'usuari clicar a iniciar partida del menú principal i se li mostra una pantalla de selecció de nivell. - L'usuari acaba un nivell i clica sobre anar al menú i se li mostra el menú de selecció de nivell.

Requeriments no Funcionals

A continuació presentaré els requeriments no funcionals del videojoc:

Identificació: RNF01	
Títol:	Compatibilitat Sistema Operatiu
Descripció:	El joc haurà de funcionar en Linux, Mac i Windows.
Prioritat:	2
Validació:	- Executar el joc a Linux i haurà de funcionar. - Executar el joc a Windows i haurà de funcionar. - Executar el joc a Mac i haurà de funcionar.

Identificació: RNF02	
Títol:	Compatibilitat diferents Resolucions
Descripció:	El joc haurà de mantenir el format visual en diferents resolucions.
Prioritat:	3
Validació:	- Executar el joc en diferents resolucions - Canviar la resolució mentre el joc s'està executant.

Identificació: RNF03	
Títol:	Els texts han de ser llegibles
Descripció:	Els texts que apareguin al joc han de ser llegibles i entenedors-.
Prioritat:	3
Validació:	- Mirar que tots els texts de les diferents pantalles es puguin llegir

Identificació: RNF04	
Títol:	Localització
Descripció:	El joc haurà de tenir localització en Anglès i Castellà; i s'ha de poder escollir el llenguatge del joc.
Prioritat:	2
Validació:	- Obrir el Joc i mirar que estigui ben localitzat al idioma seleccionat - Canviar del idioma seleccionat a l'altre i observar com es canvia l'idioma.

Identificació: RNF05	
Títol:	Interfície intuïtiva i fàcil
Descripció:	La interfície del joc ha de ser entenedora i intuïtiva. L'usuari ha de poder entendre tot el que se li mostra i accedir a totes les característiques del joc sense dificultat.
Prioritat:	3
Validació:	- L'usuari navega a través del joc sense cap problema.

Identificació: RNF06	
Títol:	Fluïdesa
Descripció:	El joc ha de córrer a 60 fps estables i sense cap problema de rendiment.
Prioritat:	4
Validació:	- L'usuari juga amb fluïdesa i sense tirons.

Casos d'ús

En aquest apartat es detallen els diferents casos d'ús que s'han definit durant la fase d'anàlisi del projecte. Els casos d'ús cobreixen les necessitats dels actors que participen e interactuen amb el videojoc.

Autors

L'únic autor d'aquest projecte és el jugador del videojoc, ja que no hi ha cap més interacció per part d'un actor ja sigui internament o externament.

Jugador: Es tracta del actor més habitual dels videojocs i alhora de utilitzar-los. Qualsevol persona pot efectuar aquest rol ja que sol requereix el joc i cap habilitat o coneixement. El joc està dirigit a jugadors de PC, ja sigui Windows, Mac o Linux.

Especificació de Casos d'Ús

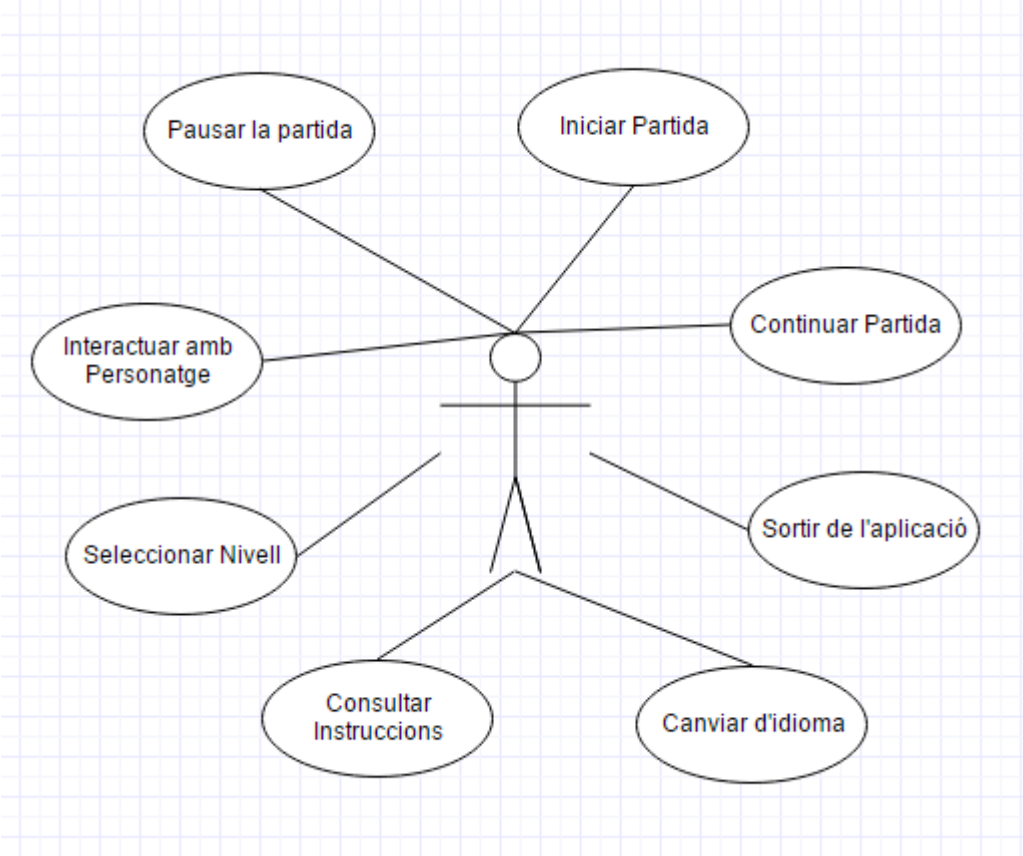
Identificador Cas d'Ús:	
Nom:	
Descripció:	
Actors:	
Precondicions:	
Postcondicions:	
Escenari:	

Cada camp significa el següent:

- *Identificador*: Codi que identifica de forma única cada cas d'ús.
- *Nom*: Nom descriptiu del cas d'ús.
- *Descripció*: Petita descripció explicativa y concisa.
- *Actors*: Actors que intervenen i/o participen.
- *Precondicions*: Accions prèvies perquè es doni el cas d'ús.
- *Postcondicions*: Accions que passaran després del cas d'ús.
- *Escenari*: Esdeveniments pas a pas del cas d'ús amb ordre determinat.

Casos d'ús del Jugador

A la il·lustració que hi ha a continuació és mostra els diferents casos d'ús en que participa el Jugador.



1. Casos d'ús del videojoc

Identificador Cas d'Ús: CU01	
Nom:	Iniciar Partida
Descripció:	El jugador podrà iniciar partida en qualsevol moment.
Actors:	Jugador
Precondicions:	Tenir el joc obert
Postcondicions:	Hi haurà una nova partida des de zero, independentment de si hi havia partida anteriorment.
Escenari:	L'usuari inicia el joc i prem el botó de Iniciar Partida

Identificador Cas d'Ús: CU02	
Nom:	Continuar Partida
Descripció:	El jugador podrà continuar una partida ja començada prèviament.
Actors:	Jugador
Precondicions:	Tenir una partida ja començada.
Postcondicions:	Continua amb la partida que ja tenia, i se li mostra els nivells que té desbloquejats.
Escenari:	L'usuari entra a l'aplicació i prem sobre el botó de Continuar Partida i se li mostren els nivells desbloquejats.

Identificador Cas d'Ús: CU03	
Nom:	Sortir de l'aplicació
Descripció:	El jugador podrà sortir de l'aplicació des de el menú principal.

Actors:	Jugador
Precondicions:	Tenir el joc obert.
Postcondicions:	L'aplicació es tanca.
Escenari:	L'usuari entra a l'aplicació i prem el botó ESC, se li mostra un submenú confirmant si vol sortir de l'aplicació i després de prémer el botó "si", l'aplicació és tanca.

Identificador Cas d'Ús: CU04	
Nom:	Canviar d'idioma
Descripció:	El jugador podrà canviar d'idioma, Anglès o Castellà, des del menú inicial.
Actors:	Jugador
Precondicions:	Tenir el joc obert.
Postcondicions:	Els texts del joc es canvien a l'idioma escollit.
Escenari:	L'usuari entra a l'aplicació, prem el botó de canvi d'idioma i l'aplicació canvia els texts a l'idioma escollit.

Identificador Cas d'Ús: CU05	
Nom:	Consultar Instruccions
Descripció:	El jugador podrà consultar les instruccions del joc.
Actors:	Jugador
Precondicions:	Tenir el joc obert.
Postcondicions:	Una pantalla amb les instruccions del joc s'ha obert.
Escenari:	L'usuari entra a l'aplicació, prem sobre el botó Instruccions i l'aplicació obra una finestra amb les instruccions.

Identificador Cas d'Ús: CU06	
Nom:	Seleccionar Nivell
Descripció:	El jugador podrà seleccionar el nivell en que vol jugar.
Actors:	Jugador
Precondicions:	Tenir el joc obert i tenir una partida carregada.
Postcondicions:	És carga i mostra el nivell que el jugador ha premut.
Escenari:	L'usuari entra a l'aplicació, continua o inicia una partida nova i escull un nivell ja desbloquejat; aquest nivell és carrega i comença la partida.

Identificador Cas d'Ús: CU07	
Nom:	Interactuar amb personatge
Descripció:	El jugador podrà interactuar amb el personatge.
Actors:	Jugador
Precondicions:	Tenir una partida carregada i estar dins d'un nivell.
Postcondicions:	El personatge és mou en conseqüència de la tecla premuda.
Escenari:	L'usuari entra a l'aplicació, selecciona un nivell i prem una tecla que faci interactuar al personatge.

Identificador Cas d'Ús: CU08	
Nom:	Pausar una partida
Descripció:	El jugador podrà pausar una partida sempre i quan estigui jugant-la.
Actors:	Jugador

Precondicions:	Tenir una partida carregada i estar dins d'un nivell.
Postcondicions:	La partida és pausa, enemics i personatges queden congelats i apareix un menú de pausa.
Escenari:	L'usuari entra a l'aplicació, selecciona un nivell, prem la tecla ESC per pausar la partida i apareix un menú de pausa.

Tecnologies actuals

En aquest apartat parlaré sobre les tecnologies actuals que hi ha per desenvolupar un videojoc, tant el motor gràfic que és la part més important ja que és la que fa córrer el joc, com l'entorn de desenvolupament que és el que et permet programar, crear les classes, etc. I per últim els diferents controls de versions per tal de guardar el teu projecte i tenir una mica de control dels fitxers.

Posteriorment explicaré el motiu de la meva tria per que fa a cada apartat comentat i quins avantatges i/o inconvenients els i veig respecte als seus "competidors".

Motor gràfic

L'elecció d'un motor gràfic no és fàcil, s'han de tenir un compte diferents aspectes com el tipus de joc que tens pensat fer, si serà amb 2D o 3D, la plataforma en que el vols llançar, si serà per a mòbils, etc.

Els teus coneixements d'aquell motor gràfic i amb el llenguatge que s'hi treballa, també seran un factor important a l'hora de la elecció i una bona tria t'ajudarà a desenvolupar més còmode, més ràpid i serà bo pel joc i la seva monitorització si és el cas.

Ara presentaré els motors gràfics que hem vaig plantejar pel desenvolupament del videojoc.

Unity

Unity és un motor gràfic fàcil d'utilitzar, una interfície simple i un gran conjunt de característiques que el fan un motor gràfic molt potent. Una de les principals característiques de Unity, és la seva integració multiplataforma, és a dir, els jocs es poden portar de manera fàcil i relativament ràpida a diferents plataformes, ja sigui Windows, Windows Phone, Android, iOS, PC, Linux, Mac, Xbox, Play Station, etc.

2. Logotip Unity

Unity treballa a alt nivell. Gràcies a la seva interfície, et permet crear objectes, escenes, muntar-te una UI (interfície d'usuari) ràpidament i moltes coses més a l'abast d'uns pocs clics.

També compta amb una gran biblioteca d'Assets que es poden descarregar directament des de la Asset Store o importar de paquets externs. Aquests paquets poden ser de diferents tipus, tant models, UI, assets o plugins, que t'ajuden i et faciliten encara més la feina.

Pot utilitzar-se junt a programes de disseny com 3DS Max, Maya o Photoshop entre d'altres. Els canvis que es realitzen als objectes creats amb aquests programes s'actualitzen automàticament a totes les instancies del joc sense tenir que reimportar a cada canvi.

El motor gràfic soporta DirectX, per Windows, OpenGL, per Linux i Mac i OpenGL ES per Android i iOS.

Els programadors poden utilitzar tres llenguatges diferents com són Javascript, C# o Boo.

Consta amb una tecnologia d'animacions anomenada Mecanim, el qual l'han desenvolupat per Unity. Inclou diferents eines de creació de màquines d'estats i manipulació de les animacions des del propi editor de Unity.

Per últim, consta amb dos versions Unity Professional, que costa des de $75/mes i Unity Personal que és totalment gratuïta.

Entre els jocs més populars desenvolupats amb Unity hi ha Crossy Road, Rust, Hearthstone, Monument Valley, entre d'altres.

Funcionament

Unity es gestiona tot a través d'un Editor. El joc és divideix en escenes com per exemple el menú principal o un nivell en concret. Dins de cada escena es poden inserir GameObjects, que són objectes amb uns components o uns altres. Cada components atorga una característica als GameObjects. Per exemple, si volem una càmera, afegim a un GameObject buit, el component Camera per tal que tot el que es trobi dins l'angle de visió de la Camera és mostri. Existixen components per a qualsevol cosa, ja sigui un text, un botó, aplicar a un objecte un Collider perquè tingui físiques, etc.

També podem aplicar lògica als objectes, afegint-li un script com a component. Així podem crear els nostres propis scripts per tal de adaptar els GameObjects a les nostres necessitats.

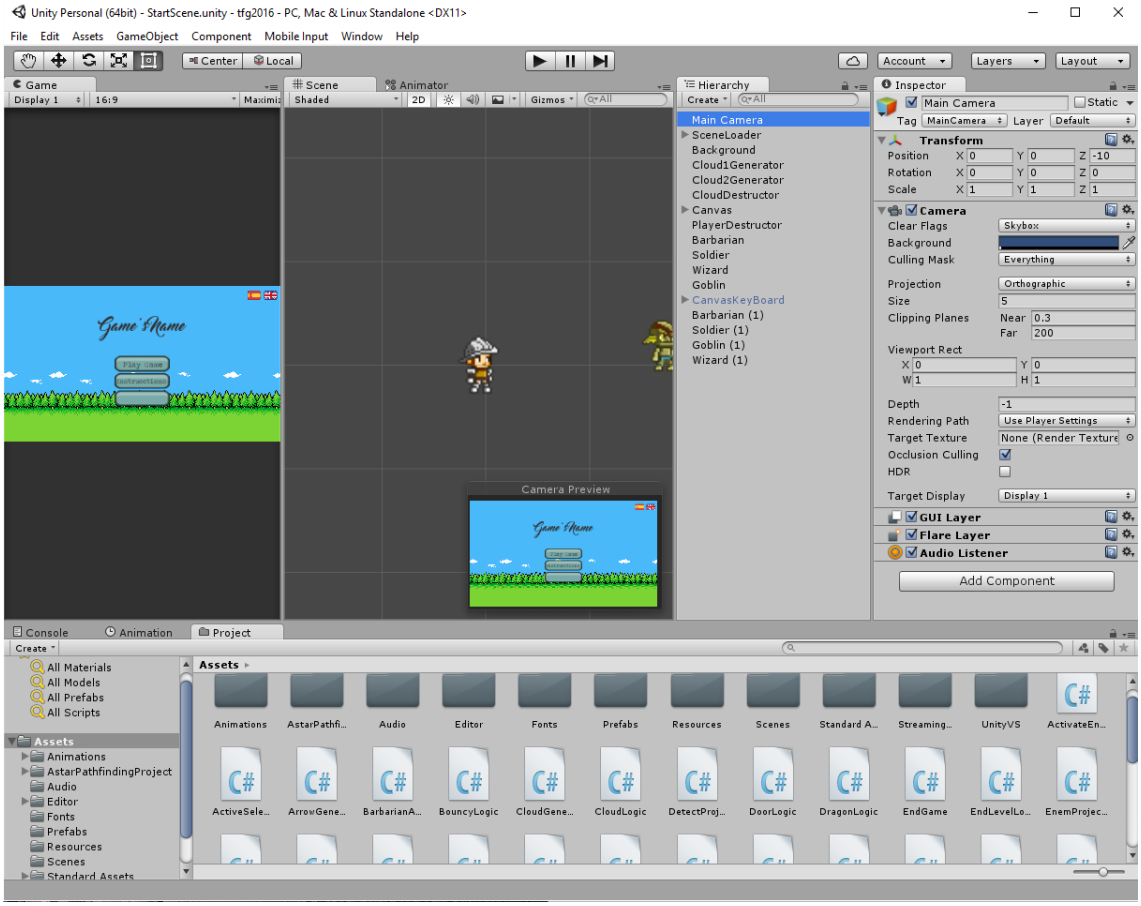
En aquest editor hi ha diferents finestres que et proporcionen diferents característiques per desenvolupar un videojoc. Les principals són:

- Game: On pots visualitzar el resultat final que es veurà quan s'estarà executant el joc.
- Scene: Que serveix per editar manualment els objectes que hi ha en una escena.
- Hierarchy: On es poden veure els Objectes que estan instanciats en aquella escena. Si al costat del nom hi ha la paraula (Clone), significa que aquell objecte ha estat creat durant l'execució del joc.
- Inspector: On, seleccionant un objecte, es poden veure els components que té i modificar els atributs d'aquests components.
- Project: En aquesta finestra tenim totes les carpetes del nostre projecte amb els arxius que conté cada una d'elles.

  També hi ha finestres que m'agrada afegir com són:

  - Console: On es pot veure tots els missatges que s'imprimeixen des de codi cap a la consola, les possibles advertències que hi pugui haver o els errors que puguin sorgir durant la programació de codi o l'execució del joc.
  - Animator: És una de les eines del Mecanim i t'ajuda a l'hora de gestionar i modificar una màquina d'
  - Animation: Et permet modificar una animació en concret.

A continuació mostraré l'Editor del Unity amb les finestres que he comentat. Les finestres es poden moure amb tota llibertat fins i tot hi ha algunes finestres que les pots tenir més d'un cop obertes.



3. Vista General de l'Editor d'Unity

Cocos2D-x

Cocos2D-x és un framework escrit en Python. És gratuït i de codi obert perelque es pot tenir el codi i adaptar-lo a les teves necessitats.

4. Logotip Cocos2D

El llenguatge de programació és, bàsicament, C++, és a dir, es necessita un bona base de C++ ja que no te entorn de disseny i s'ha de fer tot programant com el disseny de les pantalles, col·locar imatges, etc.

Cocos permet desenvolupar en diferents tecnologies mòbils com Android, iOS o Windows Phone i no permet la multiplataforma, per tant si es vol el joc en diferents plataformes, s'han de fer tants desenvolupaments com plataformes és vulgui desenvolupar.

Existeixen varies versions de Cocos2d:

Versió	**Plataforma**	**Llenguatge**
Cocos2d	Windows, OS X, Linux	Python 2.6, 2.7 o 3.3+,Objective-C
Cocos2d-x	iOS, Android, Tizen, Windows 8, Windows Phone 8, Linux, Mac OS X	C++, Lua, JavaScript
Cocos2d-ObjC	iOS, Mac OS X, Android	Objective-C, Swift
Cocos2d-html5	HTML5-ready browsers	JavaScript
Cocos2d-xna	Windows Phone 7 & 8, Windows 7 & 8, Xbox 360	C#

Entre els jocs més populars desenvolupats amb Cocos2d hi ha Hill Climb Racing, Don't Tap the White Tile, Geometry Dash o Farmville.

Unreal Engine

Unreal Engine és dels motors més potents que existeixen actualment, creat per la empresa Epic Games.

5. Logotip Unreal Engine

També treballa a alt nivell, perelque consta d'una interfície gràfica tot i que no es un software senzill de fer anar a causa del gran nombre de possibilitats de desenvolupament.

Té característiques gràfiques realment increïbles com són la capacitat de fer servir il·luminació dinàmica i un sistema de partícules capaç de gestionar milions de partícules a la vegada.

El llenguatge amb que s'ha de programar és C++ y es compatible tant amb DirectX com amb OpenGL.

Una altra característica és la multiplataforma, sent compatible tant amb PC com amb les diferents consoles actuals.

La última versió és la Unreal Engine 4 i es pot obtenir de manera gratuïta. Si els projectes desenvolupats amb Unreal Engine són comercialitzats de forma oficial, Epic Games s'enduria un 5 % dels beneficis cada trimestre a partir dels primers $3000.

Entre els jocs més populars desenvolupats amb Unreal Engine hi ha Deus Ex, Saga BioShock, Mass Effect entre molts d'altres.

Decisió presa

Finalment he decidit desenvolupar el videojoc amb el motor gràfic Unity i aquesta ha set la meva dedició per dos factors. Primer, perquè havia set amb l'únic motor gràfic que he treballat amb anterioritat i ja sabia com funcionava bastant bé. L'altre factor a sigut el llenguatge de desenvolupament. C# s'assembla molt a Java i amb Java he treballat bastant durant el grau, en canvi amb C++ no he treballat massa i perdria massa temps estudiant-me el llenguatge són per desenvolupar el joc.

A banda d'això, la interfície senzilla i fàcil d'utilitzar de Unity és perfecta per a desenvolupadors novells i ajuda molt, en canvi Cocos no té cap tipus d'interfície gràfica que el fa més complex i enrevessat. I Unreal Engine és massa complicat pel nivell de joc que he realitzat i no requeria de uns gràfics excel·lents.

Entorn de Desenvolupament Integrat (IDE)

Pel que fa a entorns de desenvolupament hi havia dos possibilitats i les comentaré a continuació.

MonoDevelop

MonoDevelop és un IDE que bé amb Unity, és a dir, és el que hi ha per defecte.

Té totes les eines que es fan servir per programar amb Unity, com és la obertura automàtica de les classes des de Unity, et deixa buscar dependències d'una funció en concret i et permet fer debugging d'una manera relativament fàcil.

És un bon entorn de desenvolupament però, a vegades por ser lent obrint el MonoDevelop el primer cop, la manera en que fa debug no és la més còmoda per al programador i a vegades és fa espès programar.

Visual Studio

Visual Studio és un IDE de Microsoft que posseeix una versió gratuïta, la Community.

També té totes les eines que té MonoDevelop, però a part també et permet crear llocs i aplicacions web, per tant va més enllà que MonoDevelop.

Per tal de poder fer servir Visual Studio amb Unity, és necessita un plugin que s'anomena VS Tools que un cop afegit al projecte de Unity en que estàs treballant, et permet enllaçar ambdós parts.

És un entorn de desenvolupament ràpid, s'obre amb certa fluïdesa i a l'hora de programar és molt còmode.

Decisió Presa

He escollit Visual Studio com a entorn de desenvolupament integrat ja que em sento molt més còmode programant amb ell i tot i que s'ha de importat un paquet és molt simple importar-lo i les millores que ofereix respecte al MonoDevelop de simplicitat i rapidesa, el fa en conjunt un millor IDE.

Servei Hosting Repositoris

El servei de hosting de repositoris o forja és un aspecte molt important també quan desenvolupes videojocs ja que tenir un lloc on poder tenir els canvis que es van fent i una copia de seguretat és imprescindible ja que normalment hi ha molts errors durant el desenvolupament i tenir una eina com aquesta et pot salvar de començar-lo de nou.

La mida del teu videojoc a l'hora de escollir un servei o un altre també és rellevant ja que no tots els servis ofereixen els mateixos preus i mides de repositori.

Ara explicaré les dos principals forges que vaig valorar.

GitHub

GitHub és el lloc per excel·lència per allotjar el codi del nostre projecte. Està basat en Git, el qual ofereix tota la funcionalitat de Git i afegeix la seva pròpia.

Ofereix una interfície gràfica basada en web i escriptori. Proporciona control d'accés i diverses característiques de col·laboració com bug tracking i wikis per cada projecte.

GitHub té dos tipus de plans; repositoris privats i comptes gratuïts. Com a limitació, sol et permeten, de forma gratuïta, una mida màxima de repositori d'1Gb i una limitació de 100 Mb per arxius.

BitBucket

BitBucket és l'altre gran servei d'allotjament basat en web i suporta tant Mercurial com Git.

Ofereix també plans comercials i gratuïts. Però a diferència de GitHub, et permeten tenir un numero limitat de repositoris privats a les comptes gratuïtes.

Pel que fa a les mides, BitBucket té dos tipus de límits, el límit suau que és de 1Gb com a mida màxima del repositori, on t'envien notificacions i missatges d'advertència i un límit dur de 2Gb com a mida màxima del repositori, on no et deixaran pujar res més fins que hi hagi menys de 2 Gb.

Decisió Presa

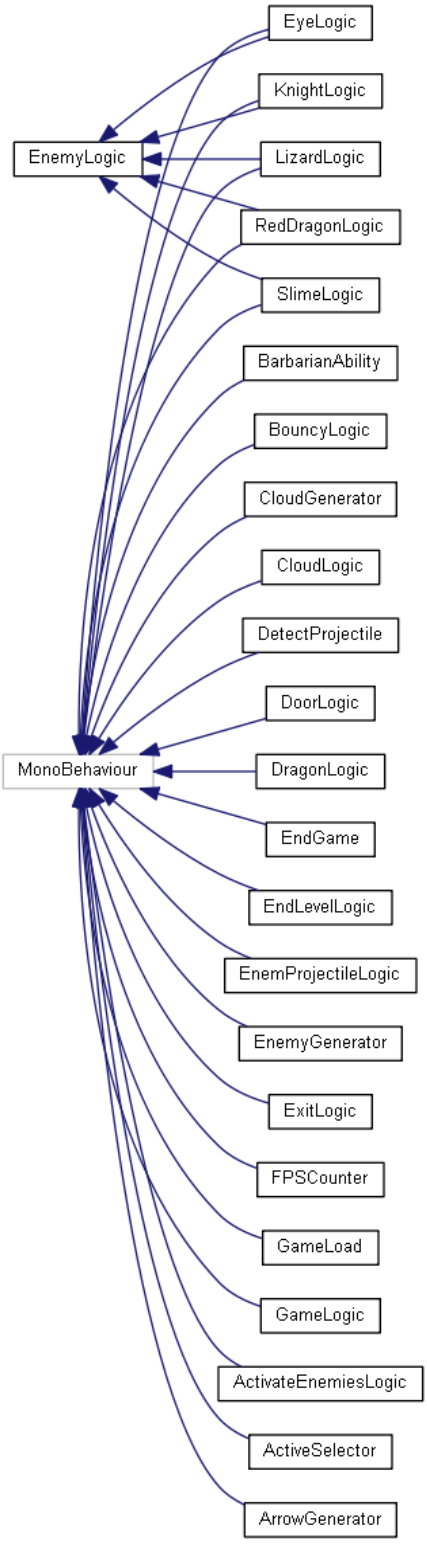
He escollit BitBucket per allotjar el codi del meu videojoc. Tot i que són bastant semblant i poden funcionar molt similar, hi ha dos aspectes que hem van fer decidir per BitBucket.

El primer és que et permet tenir una mida de repositori més gran que GitHub, fins a 2 Gb i tot i que no els utilitzaré, tens més màniga ampla per maniobrar i pujar fitxers. L'altre aspecte que hem va decantar, i que hem vaig quedar sorprès al saber-ho, és que GitHub aparentment et pot eliminar el repositori si excedeix de la mida màxima, sense previ avís. I no deu ser una cosa agradable .

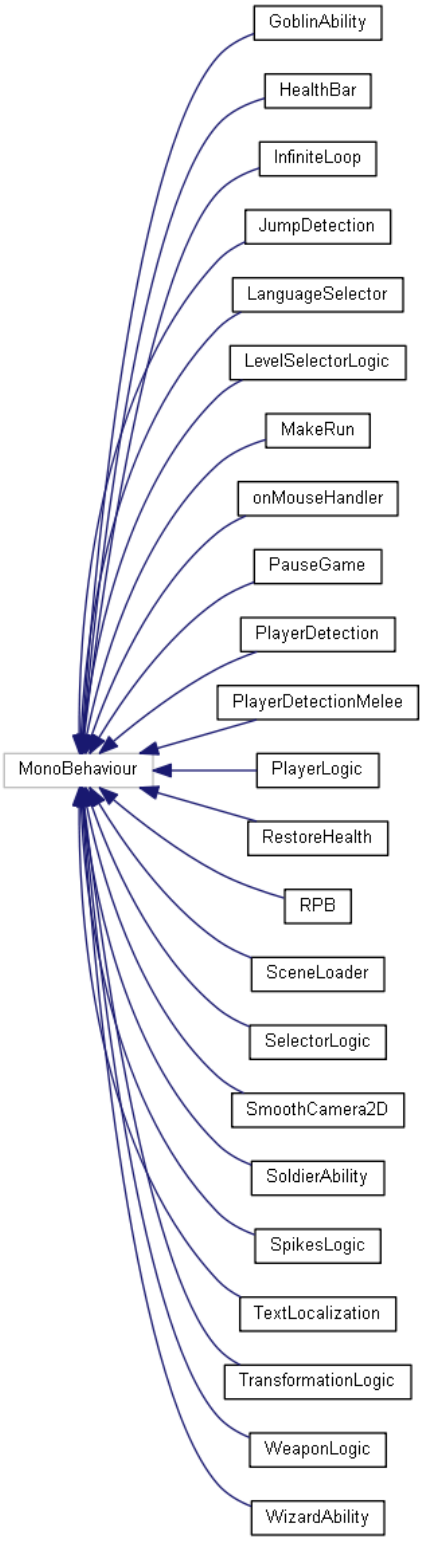
Disseny

Diagrama estàtic

En aquest apartat es mostrarà el diagrama de classes estàtic que té el projecte. Al ser un videojoc simple l'estructura no és gens complexa



7. Diagrama estàtic part1



6. Diagrama estàtic part 2

Està dividit en dos parts per tema de mida, ja que totes les classes hereten de MonoBehaviour i era massa gran, però cal destacar que en les dos imatges la classe MonoBehaviour és la mateixa.

Aquestes són totes les classes que intervenen al videojoc i les que el fan funcionar.

Com es pot observar i com ja he comentat, totes les classes hereten de MonoBehaviour que és la classe base de Unity del qual tot script deriva d'ella. Conté funcions que estan enllaçats amb events que passen durant el joc, per dir-ho d'alguna manera. Conté funcions com Start, que s'executa quan l'objecte associat a aquell script és instanciat, o Update que es crida cada frame del joc.

També podem veure que s'aplica el Patró Façana per les classes EnemyLogic, EyeLogic, KnightLogic, SlimeLogic, RedDragonLogic i LizardLogic. EnemyLogic és la interfície que serà implementada pels enemics concrets i permet treballar tots els enemics com a EnemyLogic.

Per últim, vull remarcar que el diagrama de classes ha estat creat amb un plugin de Unity, anomenat Doxygen, que et permet generar documentació API en format HTML així com també el diagrama de classes. Posteriorment s'ha representat amb el Graphviz, per poder-ho visualitzar i transformar a un fitxer de imatge png.

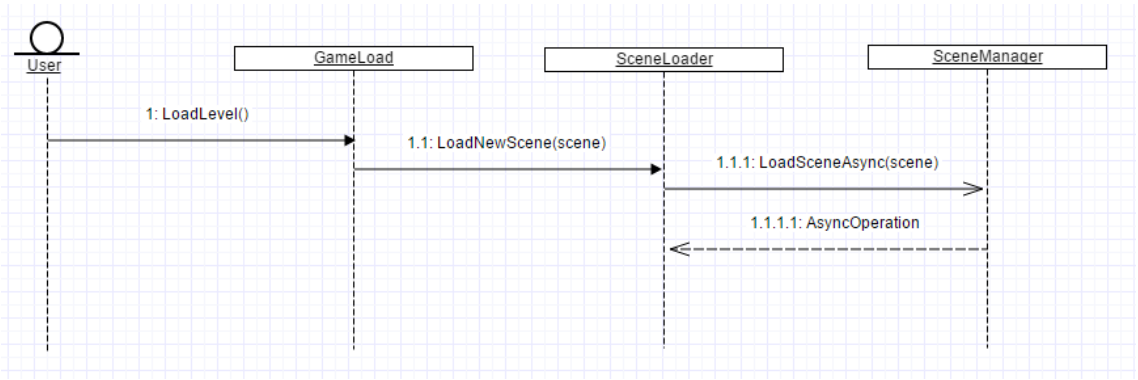
Diagrama dinàmic

A continuació és presentaran els diagrames dinàmics d'algunes de les accions més importants del joc.

Carregar escena

En aquest apartat es mostrarà com es el flux per carregar el menú de selecció de nivells. Però a partir de la crida LoadNewScene(scene) totes les escenes és carreguen de la mateixa manera.

La classe GameLoad s'encarrega de la lògica del menú principal. Quan l'usuari li dona al botó de Iniciar partida, és crida la funció LoadLevel(), aquesta funció crida la funció LoadNewScene i li passa el nom de la escena que és vol carregar, de la classe SceneLoader. Aquesta funció el que fa és instanciar un objecte que representa una carrega i dona un feedback al usuari i passa a carregar de manera asíncrona la escena des de la Classe SceneManager que és pròpia de Unity.



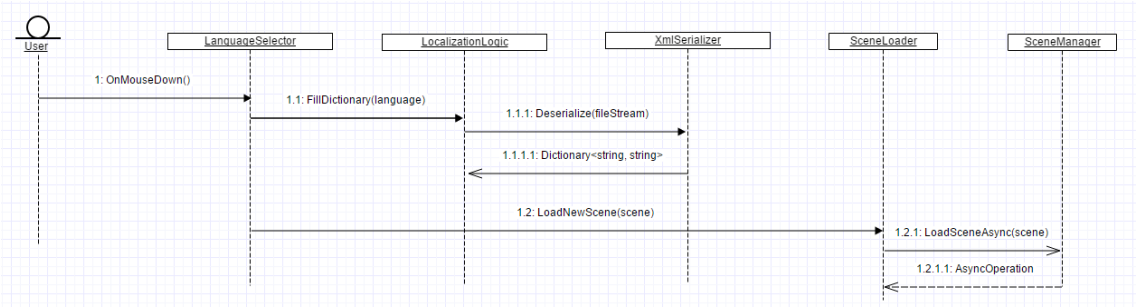
8. Diagrama dinàmic Carregar Escena

Canviar Idioma

En aquesta imatge veurem com es el procés de canviar d'idioma del joc, des del menú principal i després veurem com cada un dels GameObjects que tenen un Text que s'ha de localitzar, obtenen el nom correcte del text en l'idioma desitjatį.

Qui comença és l'usuari premen la imatge de l'idioma que vol. La classe LanguageSelector és qui registra aquest clic i passa a emplenar el diccionari amb les claus i valors corresponents a aquell idioma. Aquesta acció l'ha fa la classe LocatizationLogic a través del Deserialize passant-l'hi el fitxer del idioma. Aquesta funció retorna un diccionari que és el que es farà servir després fer buscar el valor corresponent a la clau que estem buscant.

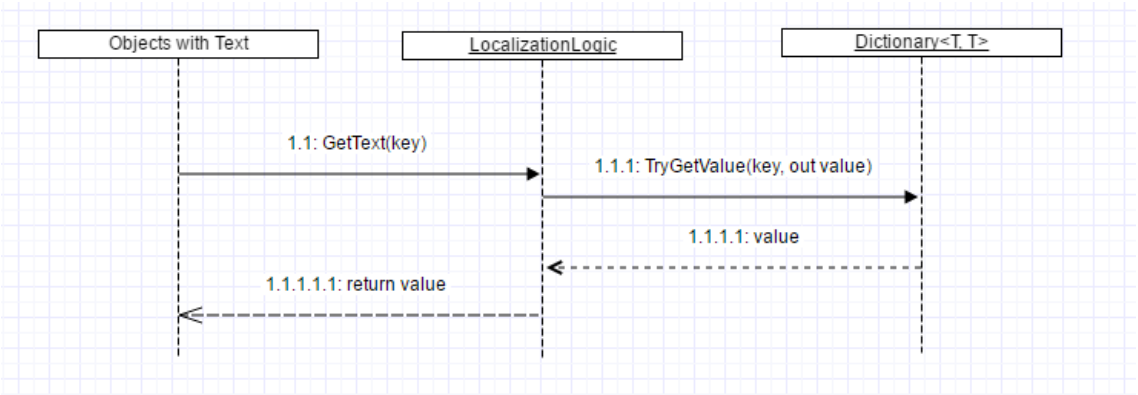
Després d'obtenir el diccionari, és carrega de nou la escena del menú principal perquè es carregui amb l'idioma correcte.



9. Diagrama dinàmic Canvi d'Idioma

Obtenir text idioma

Quan es crida la funció Start de cada GameObject que té com a Component un Text, és a dir, al instanciar-se, és crida la funció GetText(key) on key és la clau amb la qual buscarem al diccionari per obtenir el valor. Llavors és crida la funció TryGetValue on intentarà buscar la clau al diccionari generat anteriorment i si la troba, retornarà el valor a la variable value. Aquesta variable serà la que es substituirà pel camp text del GameObject.



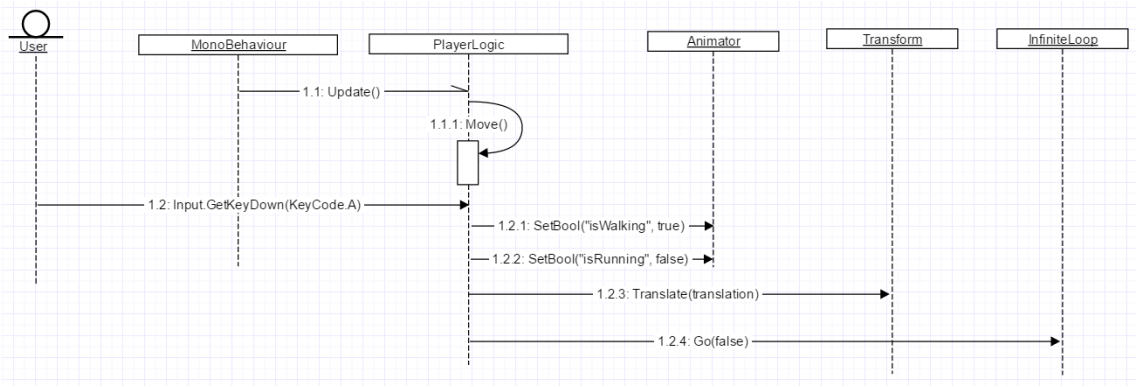
10. Diagrama dinàmic Obtenció text

Interacció Personatge

Moure Personatge

En aquesta imatge és pot veure quin es el diagrama al fer moure el personatge, endavant o enrere. El personatge té el script PlayeLogic associat, el qual a l'Update és crida la funció Move. Quan l'usuari prem la tecla A o D, el Move capta quina tecla s'ha premut i comença a efectuar els passos perquè el personatge és mogui i s'apliqui l'animació corresponent. També fa que el fons es mogui en la direcció que es mou el personatge.

Un diagrama semblant seria si fem córrer el personatge. Simplement hauríem de prémer la tecla Shift i fer que la variable isRunning sigui true.

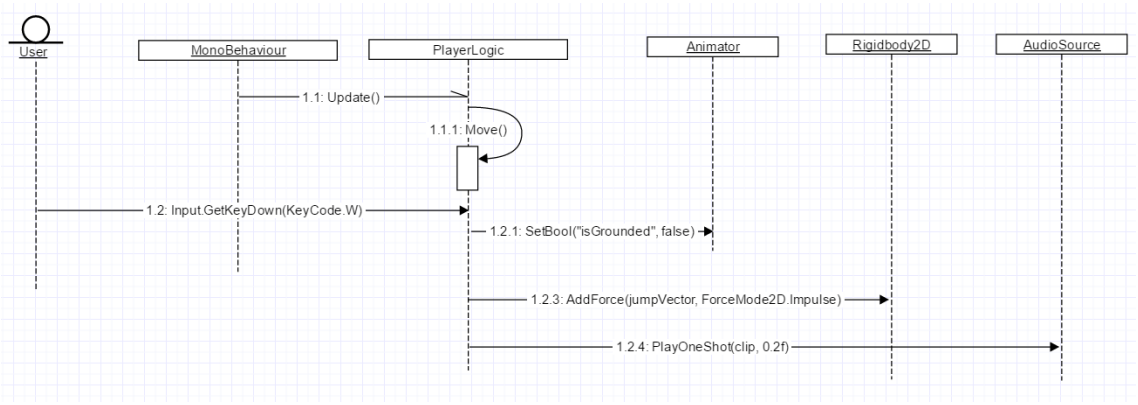


11. Diagrama dinàmic Moure Personatge

Saltar

L'acció de saltar del personatge és semblant a la de moure el personatge.

El Move es crida cada frame per tal de detectar quan premem la tecla de saltar, la W o el Espai. Quan això passa és fica la variable isGrounded, per l'animació, a true i s'aplica una força vertical al GameObjet a través del component RigidBody2D. Per últim, es crida la funció PlayOneShot que reprodueix un so de saltar cada cop que es realitza l'acció.

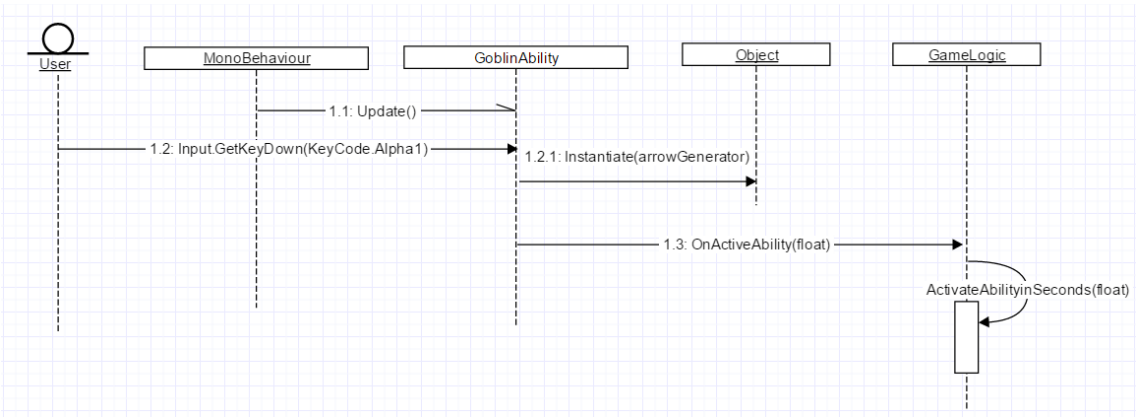


12. Diagarma estàtic salt personatge

Habilitat Especial

Pel que fa a l'habilitat especial dels personatges, sols mostraré la del Goblin, ja que a comparació de la resta, solament canvia la lògica concreta de cada habilitat, que serà explicada més endavant, però el que envolta a questa lògica és exactament igual.

Cada personatge té un scripts anomenat "nomPersonatge"Ability que capta quan l'usuari prem la tecla 1. Quan l'usuari la prem és fica en marxa la lògica concreta, que en el cas del Goblin és una pluja de fletxes. Alhora, també és fica en funcionament un compte enrere de la classe GameLogic perquè al cap de X segons s'activi un altre cop l'habilitat i es pugui tornar a fer servir.



13. Diagrama estàtic Habilitat Personatge

Videojoc

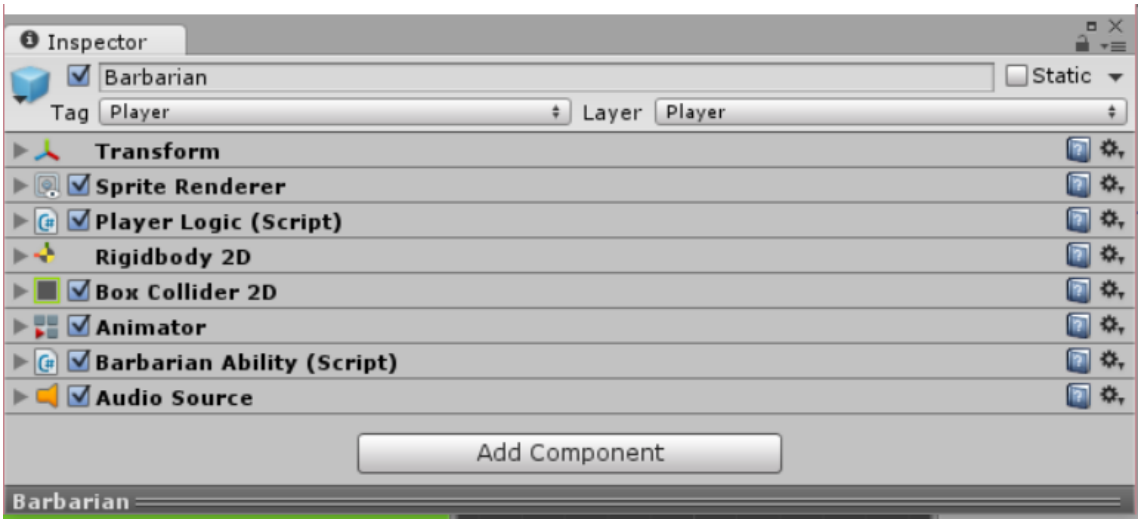
A continuació explicaré primer des d'una perspectiva més general i després d'una més especifica que forma el joc i el seu funcionament.

El videojoc que he desenvolupat és un joc per a Ordenador, exactament Windows, Mac i Linux. La temàtica del joc és plataformes, és a dir, guiar un avatar saltant entre plataformes suspeses a l'aire o saltant obstacles per avançar en la partida, en format 2D i amb un estil que gira al voltant del Pixel Art. Hi ha quatre tipus de personatges, que tenen característiques diferents i amb una habilitat especial cada un d'ells i s'enfronten a enemics amb actituds diferents envers el personatge que intenten matar.

Jugadors

Com he comentat, hi ha quatre tipus de personatges diferents que es poden anar intercanviant durant la partida i ara els aniré a descriure amb més detall.

Aquests són els components comuns que tenen els personatges i seran explicats amb més detall a continuació. El component Barbarian Ability és únic del personatge Barbarian i cada personatge té el seu propi script d'habilitat.

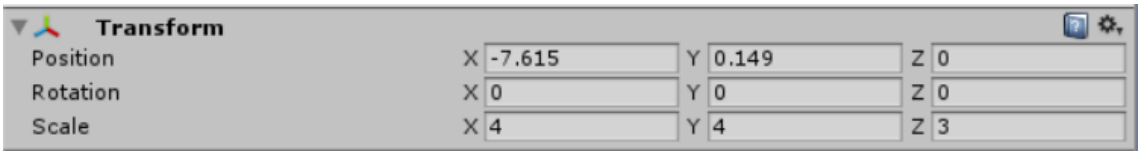


14. Components Personatge

Components

Transform

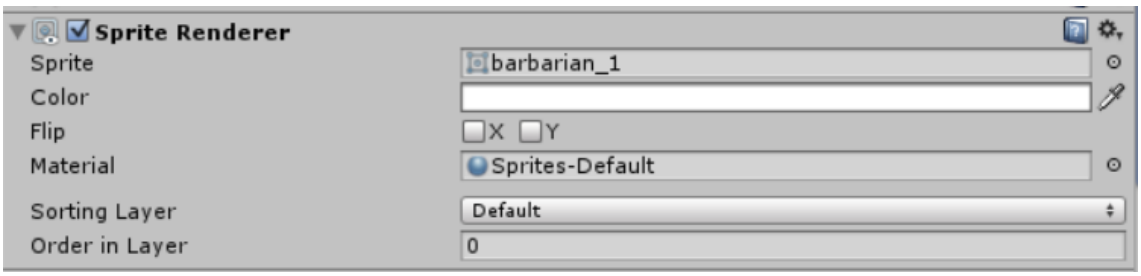
Aquest components està quasi per defecte en tots els Gameobjects ja que es el que s'encarrega de establir una posició x, y i z en el món, la escala que té i també la seva rotació. Es pot fer servir per moure un gameObject movent la seva posició.



15. Component Transform

Sprite Renderer

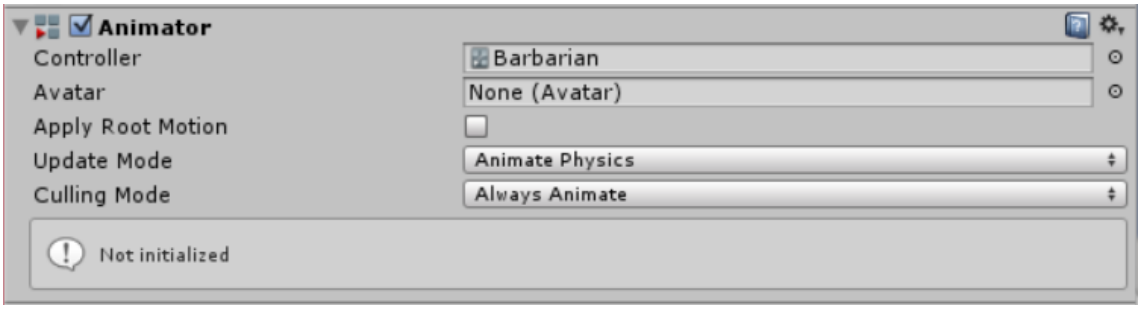
Aquest component serveix per afegir-li un sprite al gameObject. En aquest cas la imatge és la barbarian_1.



16. Component Sprite Renderer

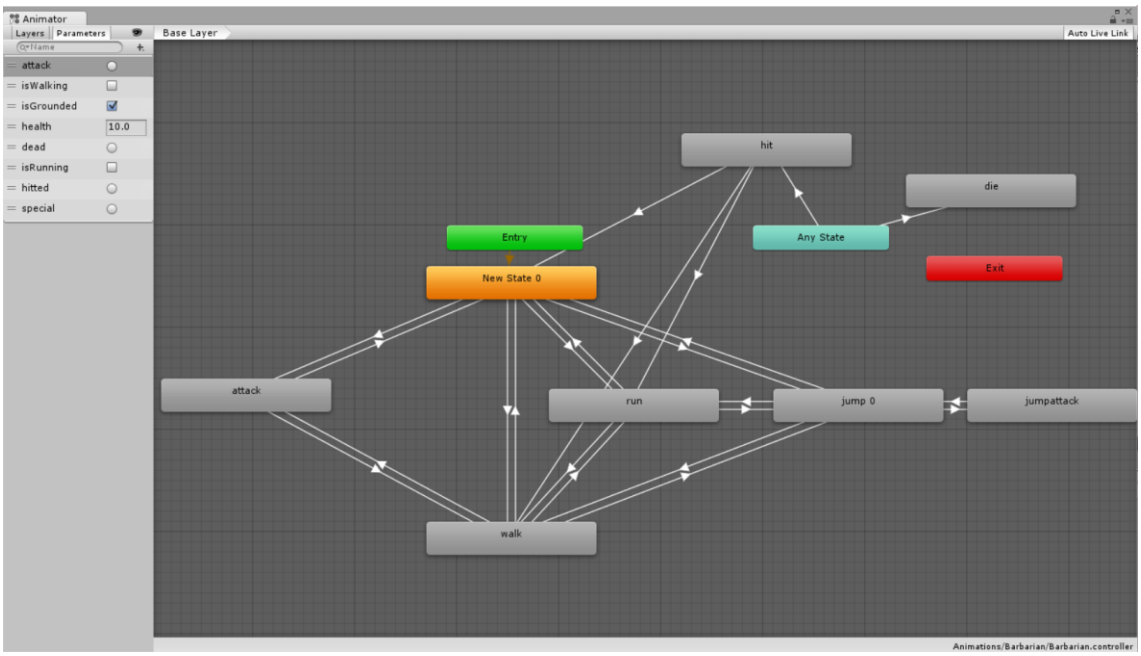
Animator

Aquest component conté la maquina d'estat amb les animacions, com se relacionen i les seves transicions. A través de variables, float, bool int o trigger és pot anar canviant d'un estat i canviar, així, les animacions.



17. Component Animator

L'arbre d'estats és el mecanisme pel qual un gameObject pot anar canviant d'animacions, ja sigui a través del temps o a través de variables i té la següent forma.



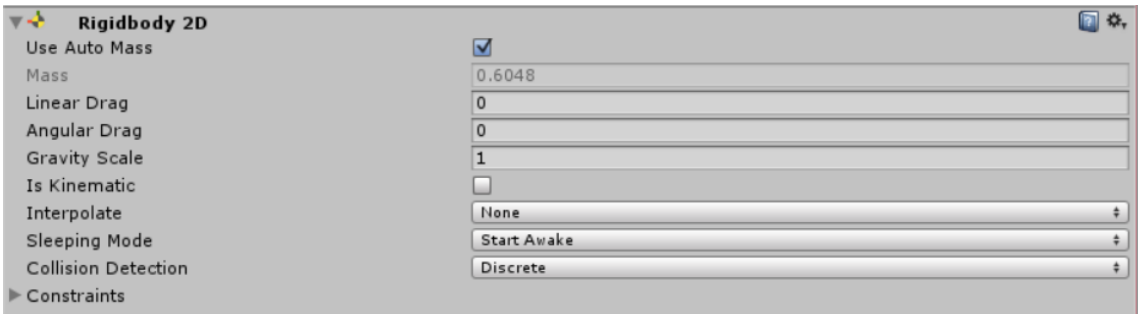
18. Màquina d'estats del Barbarian

Com podem observar, cada requadre és un estat diferent i normalment s'associa a una animació. Les fletxes simbolitzen el flux entre animacions i estan determinats per les variables que trobem a l'esquerra. La transició entre animacions pot esta determinat per cap, una o més variables.

El punt d'entrada és el requadre de color verd, el requadre taronja és el que s'executa per defecte quan s'instància l'objecte associat a aquell animator.

RigidBody2D

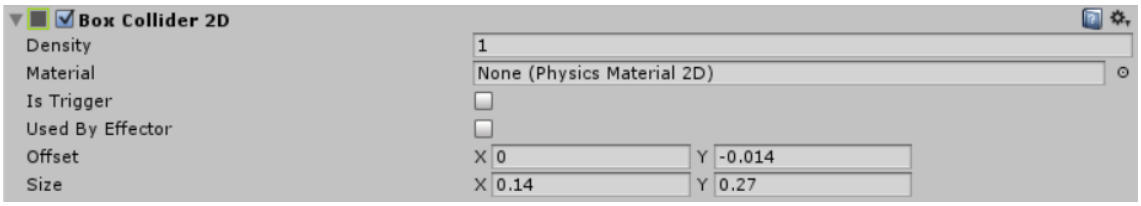
Aquest component representa els components físics d'un objecte 2D. Conté variables com la velocitat angular, la gravetat, la velocitat, etc. Afegint aquest component a un objecte, serà afectat per la gravetat o se li podrà aplicar forces.



19. Component RigidBody2D

BoxCollider2D

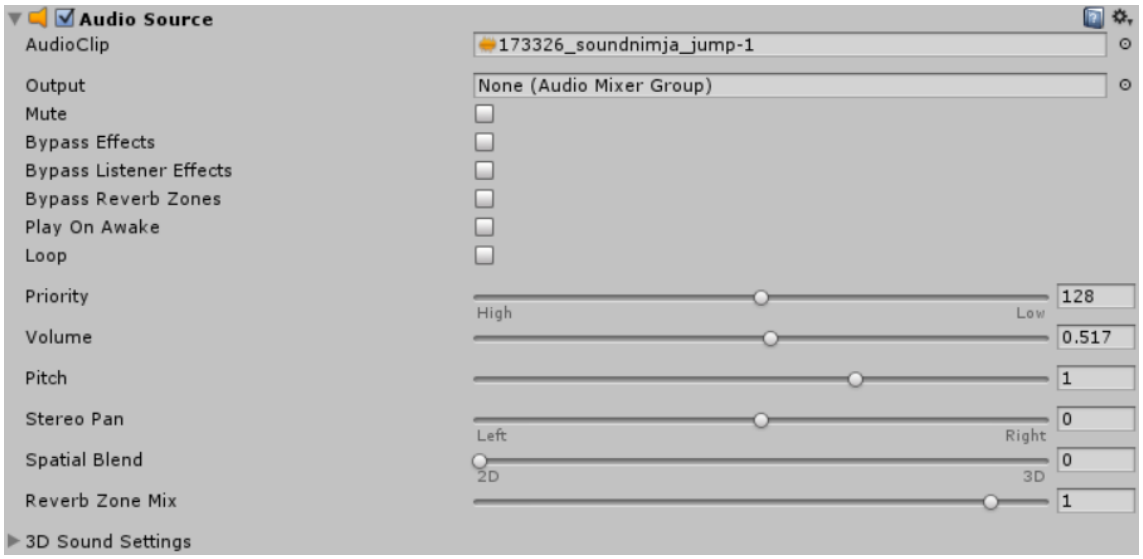
Aquest component és un collider rectangular. El collider permet les col·lisions entre objectes que tinguin colliders. Hi ha dos maneres de detectar una col·lisió. Si la variable isTrigger està desactivada, la col·lisió es detectarà amb la funció OnCollisionEnter2D. En canvi, si la variable isTrigger està activada, la funció que ho detectarà serà al OnTriggerEnter2D.



20. Component BoxCollider2D

AudioSource

Aquest component s'utilitza per afegir-hi un so al objecte i poder-lo reproduir des d'un script. El so que és reproduirà és el que es guarda a la variable AudioClip.



21. Component AudioSource

PlayerLogic

Aquest component serà explicat amb més detall a continuació. PlayerLogic és el script que controla tot el personatge i la seva implementació.

Accions

Ara es descriuran les accions que implementa PlayerLogic. Aquestes explicacions es poden acompanyar amb codi per tal de ajudar la seva comprensió.

Moviment

El moviment del personatge és gestiona des de la funció Move() que és crida a la funció Update(). Update és una funció que hereta de MonoBehaviour que s'executa cada frame del joc. La funció Move controla els quatre estats possibles del moviment com són endavant, enrere, saltar i quedar-se quiet.

La funció s'estructura en quatre parts.

En aquesta primera part, es gestiona el salt del personatge. Per tal de fer el salt, l'usuari ha d'haver premut la telca W o l'espai i el personatge ha d'estar tocant una plataforma, si aquests dos esdeveniments passen, s'aplica una força vertical al component Rigidbody2D del personatge. Després s'executa un so des del component AudioSource,

```
if (isGrounded && (Input.GetKeyDown(KeyCode.W) || (Input.GetKeyDown(KeyCode.Space))))
{
    GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpHeight), ForceMode2D.Impulse);
    this.GetComponent<AudioSource>().PlayOneShot(this.GetComponent<AudioSource>().clip, 0.2f);
}
```

22. Control Salt - Move()

Les següents dos parts són de desplaçament tant dreta com esquerra. Explicaré solament un dels casos, ja que el contrari és exactament igual però canviant la tecla que es prem i la direcció en que es moure el personatge i el fons.

```
//Move Right
if (Input.GetKey(KeyCode.D))
{
    if (shiftPressed)
    {
        animator.SetBool("isWalking", false);
        animator.SetBool("isRunning", true);
        horizontalSpeed = horizontalRunSpeed;

    }
    else
    {
        animator.SetBool("isWalking", true);
        animator.SetBool("isRunning", false);
        horizontalSpeed = horizontalOriginalSpeed;
    }

    FindObjectOfType<InfiniteLoop>().Go(true);
    direction = Direction.Right;
    GameLogic.direction = GameLogic.Direction.Right;
    SetDirectionCheckers(0.3312f, -0.326f);
    transform.Translate(Vector2.right * horizontalSpeed * Time.deltaTime);
    transform.eulerAngles = new Vector2(0, 0);

}
```

23. Control Move Right - Move()

Hi ha dos tipus de moviments, el de caminar i el de córrer. Aquesta diferència es controla amb la variable shiftpressed, true quan el shift està premut i false quan no. Depenent de la variable, s'executa l'animació de córrer o la de caminar. També és mou el fons cap a la direcció correcta i es modifica el transform, que es qui controla la posició, mida i rotació del GameObject per tal que avanci cap a la direcció desitjada.

La quarta part és la que controla que les animacions es pari i el personatge quedi quiet al lloc sense moure. S'executa quan no es prem ni la A ni la D i es fiquen les variables del Animator a false, perquè passi a l'animació de idle.

```
//Stop Animation
if (!(Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.A)))
{
    animator.SetBool("isWalking", false);
    animator.SetBool("isRunning", false);
}
```

24. Control No moviment - Move()

Atacar

La lògica de l'atac està partida. En aquest script solament hi ha el de atac a distància com és el cas del Goblin i el Wizard.

Llençar projectil

La lògica de llençar un projectil es basa simplement a instanciar un objecte, en aquest cas o una fletxa o una bola de foc i indicar-l'hi la direcció i rotació que es vol. La lògica del propi projectil està afegida com un component al propi projectil. I es aquest script qui s'encarrega de fer mal si topa amb un enemic o destruir-se si xoca amb una plataforma.

```
public void ThrowProjectile()
{
    GameObject m_projectile;
    if (direction == Direction.Left)
    {
        //Wizard
        if(this.gameObject.name.Contains("Wizard"))
        {
            m_projectile = Instantiate(projectile.gameObject, transform.position - new Vector3(1f, -0.49f), transform.rotation) as GameObject;
        }
        //Goblin
        else
        {
            m_projectile = Instantiate(projectile.gameObject, transform.position - new Vector3(0.558f, -0.03100002f), transform.rotation) as GameObject;
        }
        m_projectile.GetComponent<Rigidbody2D>().velocity = transform.TransformDirection(-Vector2.left * 10);
        Destroy(m_projectile, 3f);
    }
    else
    {
        //Wizard
        if (this.gameObject.name.Contains("Wizard"))
        {
            m_projectile = Instantiate(projectile.gameObject, transform.position + new Vector3(1f, 0.49f), transform.rotation) as GameObject;
        }
        //Goblin
        else
        {
            m_projectile = Instantiate(projectile.gameObject, transform.position + new Vector3(0.558f, 0.03100002f), transform.rotation) as GameObject;
        }
        m_projectile.GetComponent<Rigidbody2D>().velocity = transform.TransformDirection(Vector2.right * 10);
        Destroy(m_projectile, 3f);
    }
}
```

25. Funció vella ThrowProjectile()

Aquesta és la versió original de la funció. Està molt poc optimitzada i conté codi repetitiu, perelque vaig intentar optimitzar aquesta funció perquè el codi no fos repetitiu i sigues més entenedora i aquesta és la versió reduïda.

```
public void ThrowProjectile()
{
    //True: Right
    //False: Left
    int intDirection;
    Vector2 vectorDirection = new Vector2();
    GameObject m_projectile;

    if (direction == Direction.Left)
    {
        intDirection = -1;
        vectorDirection = -Vector2.left * 10;
    }
    else
    {
        intDirection = 1;
        vectorDirection = Vector2.right * 10;
    }

    //Wizard
    if (this.gameObject.name.Contains("Wizard"))
    {
        m_projectile = Instantiate(projectile.gameObject, transform.position + (intDirection*new Vector3(1f, (intDirection*0.49f))), transform.rotation) as GameObject;
    }
    //Goblin
    else
    {
        m_projectile = Instantiate(projectile.gameObject, transform.position + (intDirection*new Vector3(0.558f, (intDirection*0.03100002f))), transform.rotation) as GameObject;
    }
    m_projectile.GetComponent<Rigidbody2D>().velocity = transform.TransformDirection(vectorDirection);
    Destroy(m_projectile, 3f);
}
```

26. Funció nova ThrowProjectile()

Cos a Cos

La lògica del atac cos a cos es troba en un gameObject, fill dels personatges, anomenat Weapon tant en el Barbarian com al Soldier. Aquest gameObject activa el Component Collider a l'arma i si aquest Collider xoca amb un enemic li aplica el dany. Com aquest script també l'he fet servir per les armes dels enemics, a l'hora de la col·lisió es diferencia entre si col·lisiona amb un enemic o amb el personatge.

```
void OnTriggerEnter2D(Collider2D col)
{
    switch (target)
    {
        case "Enemy":
            if (col.gameObject.tag == target)
            {
                //Change for each Player
                GameLogic.AddScore(5);
                col.gameObject.GetComponent<EnemyLogic>().HurtEnemy(damageWeapon);
            }
            break;

        case "Player":
            if (col.gameObject.tag == target)
            {
                OnHurt(damageWeapon);
            }
            break;

        default:
            break;
    }
}
```

27. Gestió dany arma

Rebre Mal

Aquesta funció és crida cada cop que un enemic ataca i el seu atac col·lisiona amb el personatge. Està fet amb un sistema de events, que té com un símil al patró observador. No es exactament un patró observador ja que en els events coneixes el destinatari de qui has de avisar quan passa l'acció concreta i en canvi al patró observador no coneixes als observadors. Tot i això la idea es bastant semblant. És subscriu la acció desencadenant amb la de rebre mal, així sempre que es cridi el desencadenant, es cridarà la de rebre mal.

A aquesta funció se li passa un paràmetre del tipus float, que serà el mal pur. Aquest valor no es el real ja que cada jugador té armadura i és te que ajustar el valor de l'armadura amb el que s'ha fet. Per exemple, si tenim 50 d'armadura i el mal que ens fan es 20, el mal final serà de 10, perquè els altres 10 l'absorbeix l'armadura.

Se li resta a la vida que tenia actualment el personatge i s'actualitza la barra de vida. Al final hi ha un efecte de transparència cada cop que el personatge rep mal, tenint així una especia de feedback. La comprovació que es pot observar és perquè l'habilitat especial del Barbarian no és té que veure afectada per aquest efecte.

```
public void TakeDmg(float v)
{
    float dmg = ((float)(100 - this.armor) / (100)) * v;
    this.playerHealth -= dmg;

    if (this.playerHealth <= 0f)
    {
        this.playerHealth = 0f;
    }

    SetBarHealth(this.playerHealth / this.maxplayerHealth);

    //animator.SetTrigger("hitted");
    if(!(this.GetComponent<BarbarianAbility>() != null && this.GetComponent<BarbarianAbility>().isBerserked))
    {
        StartCoroutine(TransparentEffect());
    }
}
```

28. Funció TakeDmg(float)

```
EnemProjectileLogic.OnHurt += TakeDmg;
```

29. Exemple subscripció event

Barbarian

El primer personatge és el bàrbar, un home despietat, on amb el seu tors nu lluita contra els enemics amb la seva maça, feta pels propis deus Nòrdics. El seu atac cos a cos i d'una força ferotge.

Les seves característiques són les següents:

- *Dany Atac*: 45
- *Vida*: 100
- *Armadura*: 30
- *Duració Habilitat Especial*:5 segons
- *Recarrega Habilitat Especial*: 40 segons

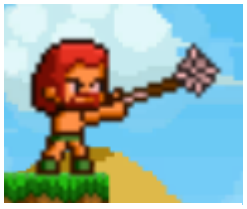
31. Asset Barbarian

30. Asset Barbarian atacant

Habilitat Especial

El bàrbar entra en mode berserker, incrementant les seves habilitats i el to de la seva pell.

El mode berserker dura 5 segons, i durant aquets 5 segons és quasi bé invencible. La seva armadura incrementa fins les 80 unitats i el dany de la seva arma també fins a les 80 unitats. El color de la seva pell es torna d'un color roig, simbolitzant els guerrers víkings que lluitaven seminus coberts de sang.



32. Mode Barbarian atacant

Goblin

El segon personatge és un follet, una criatura màgica de forma hominoide que ha aprés a lluitar i s'ha llençat a l'aventura. La seva arma és una ballesta que va heretar de la seva família, una família noble.

Les seves característiques són les següents:

- *Dany Atac*: 35
- *Vida*: 100
- *Armadura*: 20
- *Duració Habilitat Especial*:1.5 segons
- *Recarrega Habilitat Especial*: 20 segons

34. Asset Goblin

33. Asset Goblin disparant

Habilitat Especial

El follet conjura un encanteri, on invoca una pluja de fletxes davant seu que acaba amb els enemics amb que impacten. La pluja de fletxes dura 1.5 segons amb un rati de 0.13 segons, on cauen un total de 11 fletxes.

36.Pluja de Fletxes 1

35. Pluja de Fletxes 2

38. Pluja de Fletxes 3

37. Pluja de Fletxes 4

Soldier

El tercer personatge és el soldat, un guerrer entrenat des de ben petit per ser a la elit de la guàrdia i protegir al seu rei. Vestit amb la millor armadura, l'escut de la família reial i una espasa d'acer. El seu atac cos a cos no deixa indiferent a cap enemic i es que ningú mai l'ha derrotat. El seu lema és. "El millor atac és una bona defensa".

Les seves característiques són les següents:

- *Dany Atac*: 35
- *Vida*: 100
- *Armadura*: 60
- *Duració Habilitat Especial*:5 segons
- *Recarrega Habilitat Especial*: 35 segons

40. Asset Soldier

39. Asset Soldier atacant

Habilitat especial

L'habilitat especial del soldat consisteix en espantar al seus enemics. Crida tant fort i tant greu, que la defensa i el atac dels enemics baixa considerablement fent-los més dèbils al seu atac que encara és més mortal.

En un radi equivalent i inferior a la càmera, és a dir, el que renderitza la càmera, la defensa dels enemics baixa un 30% i l'atac baixa un 20%. El color de la pell dels enemics agafa un color blavós simbolitzant una baixada de les defenses del seu cos.

42.Abans Habilitat Soldier

41. Habilitat Soldier 1

44. Habilitat Soldier 2

43. Habilitat Soldier 3

Wizard

El quart i últim personatge és un mag de capa blava. Ha après a dominar la màgia a un nivell superior a la resta que li permet llençar boles de foc pel seu bastó. Els seus atacs a distància són imparables. La seva màgia no coneix límits.

Les seves característiques són les següents:

- *Dany Atac*: 30
- *Vida*: 100
- *Armadura*: 30
- *Duració Habilitat Especial*:6 segons
- *Recarrega Habilitat Especial*: 50 segons

46. Asset Wizard

45. Asset Wizard atacant

Habilitat Especial

El mag invoca un Drac Verd. Un gran mestre, li va ensenyar a transformar-se amb un gran drac verd. Aquest drac té l'habilitat de volar i durant 5 segons és pràcticament invencible.

Les característiques del drac són les següents:

- *Dany Atac*: 50
- *Vida*: 100
- *Armadura*: 90

48. Asset Drac

47. Asset Drac atacant

Enemics

En aquest apartat s'explicarà els enemics actuals del joc, la seva estructura i les seves habilitats. Hi ha una interfície comuna pels enemics, anomenada EnemyLogic que encapsula les funcions bàsiques de tots els enemics que són Atacar, Rebre mal i el Moviment. Ho vaig fer d'aquesta manera perquè en algun moment concret m'interessa tractar als enemics com a EnemyLogic i no per cada enemic concret. Hi ha cinc tipus d'enemics amb característiques i intel·ligències diferents envers el personatge i ara les presentaré.

Lizard

El llangardaix és un enemic petit amb un atac a distància potent i que et dispara allí on estiguis. El seu moviment no es ni ràpid ni lent. Va saltant per la plataforma fins que et localitza i activa el mode atac.

49. Asset Lizard

Moviment

Com he comentat, el seu moviment és constant en una plataforma a través del seu component Rigidbody2D. És mourà sempre i quan no detecti al enemic.

Atac

Per detectar el personatge i poder-lo atacar, el lizard té un collider que en quan el personatge entra en aquest collider es gira cap a la direcció que l'ha detectat i comença a executar-se la funció Attack. Com podem veure comprova si pot disparar i cada 0.5 segons s'instància un projectil cap a la direcció en que es troba el personatge.

```
private void Attack()
{
    if(!isdead && GetComponentInChildren<PlayerDetection>().canShoot)
    {
        if (m_animator.GetBool("canShoot"))
        {
            Instantiate(fireBall, this.transform.position, this.transform.rotation);
        }

    }
}
```

50. Funció Attack() – Lizard

Intel·ligència Artificial

La intel·ligència artificial d'aquest enemic és molt simple i com ja he comentat en apartats anteriors el sistema és el següent. El lizard és va movent per la plataforma fins que el personatge entra al collider, com un rang de visió, que té per detectar el personatge i un cop dins començar a atacar, si l'usuari surt d'aquest camp de visió el lizard deixa d'atacar i torna a passar al mode de caminar. El lizard segueix un tot moment la mirada al personatge, és a dir, si el lizard esta mirant a la esquerra i el collider del lizard el detecta a la dreta, el lizard és gira i comença a atacar cap on l'ha detectat.

Slime

El slime és una criatura molt petita i llefiscosa que no es preocupa per ningú i fa la seva vida senzilla i normal.

51. Asset Slime

Moviment

Aquest llimac és mou d'un costat a l'altre de la plataforma sense parar per res ni ningú. Com és pot observar la lògica de moviment és molt simple i es fa tot a través del seu component

```
public void MoveEnemy()
{
    if (shakeIt)
    {
        this.GetComponent<Rigidbody2D>().velocity = new Vector2(x*0.5f, 0);
        shakeIt = false;

    }
    else
    {
        this.GetComponent<Rigidbody2D>().velocity = new Vector2(x, 0);
    }
}
```

52. Funció MoveEnemy() - Slime

Rigidbody2D. Amb la variable shakeIt intento representar una mica el moviment d'onada, és a dir, moure ràpid el cap i com la cua el segueix a menys velocitat.

Atac

Aquest enemic no té cap tipus de atac dirigit al personatge. L'únic possible atac que te aquest llimac és ell mateix, ja que quan el personatge passa a través d'ell, rep mal.

Intel·ligència Artificial

No té cap tipus de intel·ligència ja que simplement és mou de esquerra a dreta i a la inversa constantment.

Red Dragon

El drac roig és una criatura ferotge amb un gran potencial que busca constantment al personatge per matar-lo. No escup foc i tampoc pot volar, però no el fa menus ferotge.

53. Asset RedDragon

Moviment

El moviment d'aquest drac és semblant al del Lizard. És mou a través de la plataforma fins que el personatge entra al seu collider i el persegueix incrementant la seva velocitat. Com podem observar, la lògica és simple ja que simplement es comprova si esta seguint al personatge o solament està caminant per la plataforma. En el cas que estigui dins del seu rang, incrementa la velocitat.

```
public void MoveEnemy()
{
    this.GetComponent<Rigidbody2D>().WakeUp();
    isAttacking = false;

    attackOnce = false;
    if (m_animator.GetBool("following"))
    {
        CalculateVariables();
    }
    if(distance > rangeDistance)
    {
        this.GetComponent<Rigidbody2D>().velocity = new Vector2(velocityX, 0);

    }

}
```

54. Funció MoveEnemy() - RedDragon

Atac

L'atac del drac és cos a cos, i és que quan acaba de perseguir-lo i aconsegueix atrapar-lo, el drac passa a l'atac. La funció Sleep del component RigidBody2D, el que fa és no moure's fins que el personatge no estigui fora del seu rang d'atac.

```
public void AttackEnemy()
{
    this.GetComponent<Rigidbody2D>().Sleep();
    OnHurt(20f);
}
```

55. Funció AttackEnemy - RedDragon

Intel·ligència Artificial

La seva intel·ligència artificial és una mica més complexa que la dels altres dos enemics. Quan detecta el personatge, el persegueix augmentant la seva velocitat i quan esta a una distància en que el pot atacar, para i comença a atacar. La persecució es reprèn quan el personatge esta fora del seu rang d'atac o si està ja fora del seu rand de visió, torna al mode de caminar.

Knight i Eye

Aquests dos enemics són com un de sol. Tenen una associació d'esclavatge, és a dir, el Knight és el cap i els Eyes els esclaus. Cada Knight té al seu poder un nombre d'Eyes que lluiten per ell i el defensen.

56. Assets Eyes

57. Asset Knight

Moviment

El moviment del Knight és nul, no és mou de la seva posició en cap moment. En canvi els esclaus és mouen des del moment en que el cap visualitza al personatge. El moviment dels Eyes és fa a través d'un pathfinding. Aquest pathfinding es realitza amb l'algorisme A* i una heurística euclidiana, que a través d'un plugin, s'ha col·locat una espècie de malla evitant els llocs per on el Eye no pot passar. Quan s'ha de iniciar el moviment, el Eye detecta la posició del Personatge i traça una ruta cap a ell amb els waypoints que ha de recórrer per arribar-hi. Si el personatge canvia de posició es traça una nova ruta i s'elimina la vella. Ho podem veure en la funció UpdatePath, on a cada volta calcula el camí entre ell i el personatge. Quan aquest arriba al seu objectiu, es crida la funció OnPathComplete.

Si el Knight ja no veu al personatge, fa tornar al seus esclaus cap a ell perquè el protegeixin, representat per la funció ReturnsHome, que simplement traça una ruta entre la posició actual del Eye i la del seu cap, el Knight.

```
private IEnumerator UpdatePath()
{
    if(target == null)
    {
        SearchPlayer();
    }

    //Start a new path to the target position, return the esult to the OnPathComplete method
    seeker.StartPath(this.transform.position, target.position, OnPathComplete);

    yield return new WaitForSeconds(1f/updateRate);

    StartCoroutine(UpdatePath());
}
```

58. Funció UpdatePath - Eye

```
public void ReturnsHome()
{
    target = patherBoss;

    seeker.StartPath(this.transform.position, target.position, OnPathComplete);

    StartCoroutine(UpdatePath());
}
```

59. Funció ReturnsHome() - Eye

Atac

L'atac del Knight és contundent però d'una baixa mobilitat, ja que simplement ataca si el personatge es troba davant seu. Com es pot obsevar, si la distancia entre el personatge i el Knight és més gran de 3, simplement envia els seus esclaus a atacar i si es mès petita que 3 ataca ell mateix.

```
public void AttackEnemy()
{
    if (distance > 3f)
    {
        SendSlavesAttack();
        alreadySentBack = false;
    }
    else if (distance < 3f)
    {
        //Attack himself
        alreadySentAttack = false;
    }

}
```

60. Attack Enemy - Knight

L'Atac dels Eyes no és tant un atac com més una molèstia, ja que quan arriben a contactat amb el personatge el seu atac és molt dèbil i quasi no fa mal, però molesta.

Intel·ligència Artificial

EL Knight no té cap tipus d'intel·ligència artificial per ell sol ja que simplement ataca si el te suficientment a prop. Però en conjunt, formen com una parella letal, ja que si esta suficientment lluny el Knight envia als seus esclaus perquè ataquin i molestin al personatge per quan arribi estigui ja ferit i el pugui matar amb més facilitat. La intel·ligència artificial dels Eyes és complexa ja que requereix un sistema de pathfinding per trobar el camí cap al personatge.

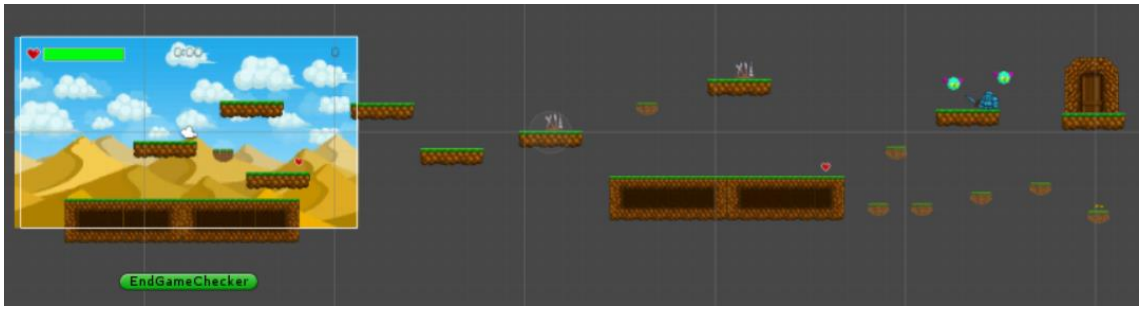
Nivells

En aquest apartat parlaré sobre el disseny pròpiament dels nivells i dels elements que formen un nivell.

El disseny dels nivells és, possiblement, una de les tasques més difícils que he realitzat i una de les tasques més difícils que hi ha a l'hora de desenvolupar un videojoc. Per això existeixen expert en disseny, altrament dit dissenyadors. Aquests dissenyadors s'encarreguen de desenvolupar el esquem, concepte i jugabilitat del joc, és a dir, anivellar tant els personatges com els elements, disseny del terreny de joc per, en definitiva, fer que el joc sigui divertit i equilibrat.

Com jo no són dissenyador ni cap dissenyador ha fet el disseny del meu joc, segurament no exprimeix tota la diversió que es podria treure i, segurament, tampoc està del tot anivellat pel que fa a les estadístiques dels personatges i enemics.

Pel que fa al disseny del nivell, he creat dos nivells i he intentat que siguin entretinguts i alhora, no s'assemblin entre ells. En aquesta imatge és pot observar la distribució dels elements en el primer nivell. A continuació s'explicarà en més detall cada un dels elements que hi pot haver en un nivell.



61. Vista general distribució Nivell 1

Fons

El fons és una mesh Renderer on la textura és el fons de cada nivell. Una Mesh és un conjunt de vèrtexs, vores i cares que descriuen una forma d'un objecte 3D, tot i que jo ho faig servir com un pla 2D. La textura que faig servir de fons està en mode Repetitiu per tal que quan acabi torni a començar i sembli un fons infinit.

Plataformes

Hi ha 3 tipus de plataformes per les quals, el personatge pot caminar i passar-se el nivell. Les plataformes no tenen cap tipus de script per gestionar la lògica ja que no tenen cap tipus de lògica. Simplement tenen un component BoxCollider2D, per tal que el personatge pugui parar a la plataforma sense caure al buit. Se l'hi ha assignat un tag anomenat Ground per tal que la malla que es fa servir pel pathfinding pugui trobar les plataformes i evitar que es pugui passar pel mig d'elles.

Com es pot observar en aquesta imatge, cada una de les plataformes conté 2 colliders als extrems per tal que els enemics detectin quan s'acaba la plataforma en que estan i donar la volta per no caure al buit.

62. Plataforma amb Colliders

Plataforma petita

Aquest és el model de plataforma petit, serveix com a interconnexió entre dos plataformes mitjanes i/o grans.

63. Asset Plataforma petita

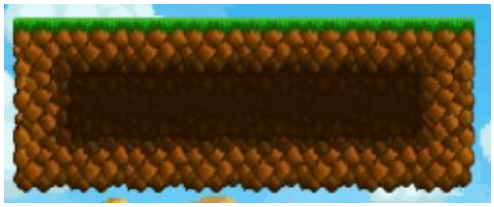
Plataforma mitjana

Aquest model és la plataforma mitjana on hi poden haver enemics o algun item.

64. Asset Plataforma mitjana

Plataforma gran

Aquest tipus de plataforma és poden ajuntar per tal de fer una plataforma més gran on enemics com el drac tenen un major potencial d'atac.



65. Asset Plataforma gran

Spikes

Les punxes són un element contra el personatge. El sistema de mal de les punxes es el següent, el primer contacte fa més mal i cada 0.1 segons que el personatge esta dins de les punxes, fa mal, però menys.

66. Asset Spikes

```
void OnTriggerEnter2D(Collider2D col)
{
    if (col.gameObject.tag == "Player")
    {
        StartCoroutine(hurtEnemy);
    }
}

private IEnumerator HurtPlayer()
{
    OnHurtPlayer(10f);
    while (true)
    {
        yield return new WaitForSeconds(0.1f);
        OnHurtPlayer(5f);
    }
```

67. Logica Mal - Spikes

Hearth

El cor és un element que et proporciona vida en moments de debilitat o després d'una gran lluita. Et restaura un 20% de la teva vida total.

68. Asset Hearth

Key and Door

L'ultim element o elements que es poden trobar en un nivell són la clau i la porta.

La clau i la porta són els elements principals per passar-te un nivell i es que si arribes a la porta podràs guanyar el nivell, sempre i quan s'hagi aconseguit la clau que està situada en una posició concreta del mapa. Hi ha 2 posicions pel que fa a la porta, tancada i oberta. Sempre estarà tancada, a no ser, que l'usuari aconsegueixi la clau i es situï a la porta. En aquell moment, la porta s'obre i prement la tecla d'acció, el nivell estarà passat.

71. Asset Door tancada

70. Asset Key

69. Asset Door oberta

Generador Enemics

Per tal de millorar el rendiment del joc i que el rati de frames del joc sigui estable quan hi ha molts enemics, vaig implementar una manera que a través d'un generador d'enemics i dos colliders situats a cada costat de la càmera, millorés aquest aspecte.

Abans d'aquesta millora, quan hi havia el Knight amb 2 o més esclaus, eyes, els frames del joc baixaven fins a 40 fps, ja que el sistema de pathfinding consumeix bastant. El rati de frames també baixava ja que estaven tots els enemics generats i tenir gameObjects fora de la pantalla però executant-se, apart de costos es ineficient.

El que vaig pensar és, ficar dos colliders a cada costat de la pantalla, allunyats del marc d'aquesta i que simplement detectessin si la càmera es mou cap a la esquerra o cap a la dreta.

També vaig ficar uns generadors d'enemics repartits pel mapa, amb un Enemic associat a cada un d'ells per tal que un d'aquells colliders l'activi i l'altre collider el desactivi. El funcionament és el següent. Si anem cap a la dreta, el collider de la dreta al col·lisionar amb els generadors, aquests fan activar o instanciar l'enemic que tinguin associat. El collider de la esquerra, en canvi, els farà desactivar. I si el personatge va cap a la esquerra, el funcionament és el contrari.

Com és pot veure en la imatge, tenim per un costat la funció de Crear o Activar un enemic, i per altra banda la funció de desactivar-lo.

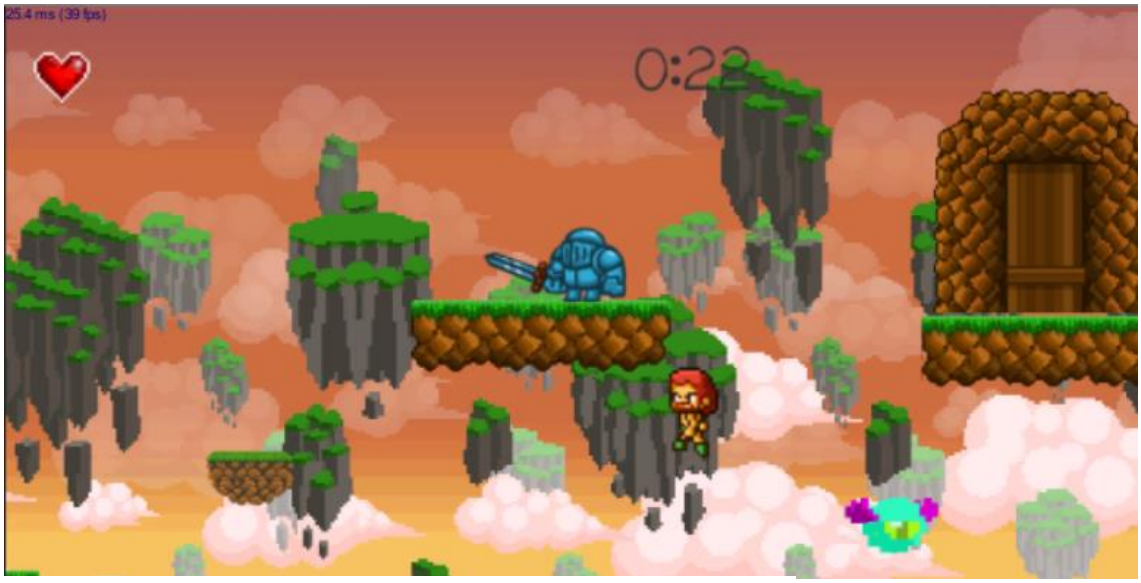
```
public void CreateActiveEnemy()
{
    if (!isGenerated)
    {
        _enemy = Instantiate(EnemytoGenerate, this.transform.position, this.transform.rotation) as GameObject;
        isGenerated = true;
    }
    else
    {
        if(_enemy != null)
            _enemy.SetActive(true);
    }
}

public void DeactiveEnemy()
{
    if (isGenerated)
    {
        if(_enemy != null)
            _enemy.SetActive(false);
    }
}
```

72. Funció Creació/Activació i Desactivació enemics

Amb aquesta millora, vaig aconseguir que el joc, quan hi haguessin el Knight i Eyes funciones a 60pfs i no hi hagués caigudes de frames. Crec que és un bon sistema ja que s'estalvia de fer crides cada frame de totes les funcions Update i FixedUpdate dels scripts de cada Enemic.

En les imatges que hi ha continuació es pot apreciar la millora en els fps de la versió vella a la nova amb els canvis corresponents.



73. FrameRate inestable sense control de Generadors

74. FrameRate estable amb control de Generadors

Sistema de Guardat

El sistema de guardat és un sistema el qual conté tots els nivells actuals que hi ha al joc, si el nivell en qüestió està desbloquejat i la màxima puntuació que ha obtingut l'usuari, 0 si encara no ha jugat.

El funcionament general és el següent, quan l'usuari obre el joc, és mira si hi ha alguna partida guardada, si no n'hi ha cap és crea una partida nova, buida on esta tot bloquejat excepte el primer nivell. Si troba alguna partida guardada, la carrega i guarda les variables a un diccionari que és d'on el joc traurà la informació després. Això pel que fa al sistema de Loading.

A l'hora de guardar una partida, és modifica el diccionari on s'ha carregar anteriorment la informació i es guarda a un fitxer.

Ara explicaré com funcionen els mètodes de guardar i carregar.

Primer de tot, s'ha creat una classe Serializable, això li diu a Unity que aquest script és pot serialitzar, és a dir, que es poden guardar totes les variables d'aquest script. Serveix per poder carregar les dades del fitxer com una classe directament. Aquesta classe la farem servir tant per carregar com per guardar les dades al diccionari.

```
[Serializable]
public class SavedData
{
    public string name;
    public int score;
    public bool unlocked;

    public SavedData(string name, int score, bool unlocked)
    {
        this.name = name;
        this.score = score;
        this.unlocked = unlocked;
    }
}
```

75. Classe SavedData

La classe s'anomena SavedData i té 3 variables.

- name: Conté el nom del nivell, tipus string.
- score: Conté la màxima puntuació obtinguda en aquell nivell, tipus int.
- unlocked: Determina si aquell nivell està bloquejat o no, tipus boolean.

Comencem per la funció de Carregar les dades des del fitxer al diccionari.

Primer de tot mirem si existeix el fitxer, és a dir, si hi ha alguna partida guardada. Les partides és guarden al directori persisentDataPath. Aquest directori és on s'espera que hi hagi la informació que es pot guardar entre execucions del joc.

Si el fitxer no existeix el diccionari s'omple amb variables buides, començant el joc de zero.

Si el fitxer si que existeix, comença un procés de deserialització per obtenir les dades. Primer creem una instància de la classe BinaryFormatter. Aquesta classe s'utilitza per serialitzar i deserialitzar un objecte en format binari. Acte seguit s'obté un FileStream, que representa una seqüencia de bytes del fitxer que volem obtenir i permet operacions d'escriptura i/o lectura tan de forma síncrona com asíncrona. Fem servir la funció deserialize de la variable BinaryFormatter i li passem com a paràmetre el FileStream, fiquem un cast per tal que ens retorni un diccionary amb clau el numero de nivell que és i com a valor, una instància de la classe SavedData amb les dades corresponents del nivell. I per acabar tanquem el FileStream.

```
public void Load()
{
    if(File.Exists(Application.persistentDataPath + "/playerInfo.dat"))
    {
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Open(Application.persistentDataPath + "/playerInfo.dat", FileMode.Open);
        DataDict = (Dictionary<int, SavedData>)bf.Deserialize(file);
        file.Close();

        Debug.Log("File Loaded from: " + Application.persistentDataPath + "/playerInfo.dat");
    }
    else
    {
        CreateEmptyDict();
    }
}

private void CreateEmptyDict()
{
    AddDict(1, new SavedData("first", 0, true));
    AddDict(2, new SavedData("second", 0, false));
    AddDict(3, new SavedData("third", 0, false));
    AddDict(4, new SavedData("fourth", 0, false));

}
```

76. Funció Load - SerializeData

La funció de guardar és la que s'encarrega de ficar el diccionari actual i substituir-lo pel que hi havia al arxiu. Primer de tot, mostraré com s'afegeix o es modifica una variable del diccionari en el que es guarda la informació i és fa amb la funció AddDict.

Aquesta funció té dos paràmetres de entrada, el número del nivell en format int, i la informació del nivell en format SavedData. Serveix tant com per actualitzar com per afegir.

Primer de tot es comprova que el nivell sigui vàlid i estigui dins dels rangs, a posteriori es comprova si el diccionari conté la clau, que voldrà dir que s'ha d'actualitzar. En aquest cas es substitueix el que hi havia per la nova variable de SavedData. Si la clau no existeix, s'afegeix el número i la data al diccionari.

```
public void AddDict(int level, SavedData data)
{
    if(level < 5 && level > 0)
    {
        if (DataDict.ContainsKey(level))
        {
            //Update Value
            DataDict[level] = data;
        }
        else
        {
            DataDict.Add(level, data);
        }
    }
}
```

77. Funció AddDict - SerializeData

Simplement la funció Save el que fa és crear una instànciade BinaryFormatter, una Instànciade la classe FileStream amb mode Obrir o Crear i cridar la funció Serialize amb el FileStream i la data que és vol emmagatzemar, en aquest cas el diccionari. Un cop serialitzat i guardar al fitxer, és tanca el FileStream i ja està.

```
public void Save()
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Open(Application.persistentDataPath + "/playerInfo.dat", FileMode.OpenOrCreate);
    bf.Serialize(file, DataDict);
    file.Close();
    Debug.Log("File Saved at: " + Application.persistentDataPath + "/playerInfo.dat");
}
```

78. Funció Save - SerializeData

Localització

La localització d'un videojoc és el procés pel qual es prepara tant el software com el hardware del videojoc per tal que pugui ser importat i venut a diferents països i/o regions. Tot i que la traducció dels texts és una gran part important, a vegades també pot implicar canviar l'art dels assets, la música, crear nous paquets, retallar el joc per les sensibilitats culturals, entre d'altres coses.

En el mes cas, solament he realitzat la traducció dels texts del joc, ja que no he cregut convenient afegir, modificar o eliminar cap altra part ni aspecte del joc. I a continuació explicaré com ho he realitzat.

Primer de tot vaig esta pensant com poder realitzar aquest procés i el que vaig pensar va ser, guardar per cada idioma que volia traduir un fitxer amb format xml, on hi haguessin parelles de valors, on una parella està formada per la clau d'una paraula i com a valor la traducció amb l'idioma del fitxer. És a dir, si tenim el fitxer english.xml, una possible parella seria, clau="door" valor="door". Per cada idioma, les claus de les paraules són les mateixes i el que canvia és el valor.

Un cop tenia el sistema dels fitxers, la idea era guardar els fitxers en una carpeta dins del projecte, que un cop feta la build aquesta carpeta també hi estigués per poder accedir-hi des del codi. Unity proporciona una carpeta anomenada StreamingAssets el qual tots els fitxers i/o carpetes dins de la mateixa, es copien a la maquina on vols fer la build i des de codi és pot accedir amb la variable Application.streamingAssetsPath. Aquesta variable era el que necessitava per poder passar els fitxers de traducció a la build final.

Llavors solament tenia que llegir del fitxer de l'idioma desitjat i guardar-ho en un diccionari per que cada text pugues accedir-hi més tard.

En les següent imatges és mostra la implementació que vaig fer per tal d'aconseguir per la Localització dels texts.

Per fer-ho vaig tenir que crear una classe anomenada item. Aquesta classe consta de dos paràmetres, id i value. Aquests paràmetres estan marcats com a [XmlAtribute] per tal que quan deserialitzi del fitxer, agafi cada paraula com una instància de la classe item i els hi assigni els valors a les variables, ja que tenen el mateix nom que al fitxer.

```
public class item
{
    [XmlAttribute]
    public string id;
    [XmlAttribute]
    public string value;
}
```

79. Classe Item

```
<?xml version="1.0" encoding="Windows-1252"?>
<items xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <item id="playGame" value="Play Game" />
  <item id="playAgain" value="Play Again" />
  <item id="exit" value="Exit" />
  <item id="mainMenu" value="Main Menu" />
  <item id="yes" value="Yes" />
  <item id="no" value="No" />
  <item id="options" value="Options" />
  <item id="instructions" value="Instructions" />
```

80. Fragment xml localització english

La funció anomenada FillDictionary(string) de la classe LocalizationLogic, que serveix per emplenar el diccionari amb les paraules de l'idioma desitjat, segueix la mateixa tònica que les

funcions que vam veure del sistema de Guardar i de Carrega, però en canvi de tenir els fitxers en format de bytes, el tenim en format xml. Com podem observar, creem una instància de la classe XmlSerializer que ens servirà per deserialitzar. Al constructor li passem diferents paràmetres, en el primer li indiquem la classe en que estan representades les dades, i en el segon, una instància de XmlRootAttribute indicant-l'hi el nom del element xml que generen i reconeixen els mètodes Serialize i Deserialize.

Com van fer anteriorment, també creem una instància de FileStream del fitxer que volem tractar. A posteriori cridem el mètode Deserialize del fitxer i fent un cast de la classe que volem obtenir. La funció ToDictionary és una funció per convertir llistes a diccionaris i el que li passo com a paràmetres és una consulta LINQ, que simplement recorre la llista, on "i" és cada element i obtenim el id i el value d'aquesta llista i el fiquem com a clau, valor.

Finalment tanquem el FileStream i ja tenim carregat al diccionari l'idioma que hem seleccionat.

```
public static void FillDictionary(string language)
{
    XmlSerializer serializer = new XmlSerializer(typeof(item[]), new XmlRootAttribute() { ElementName = "items" });

    FileStream fileStream = new FileStream(Application.streamingAssetsPath + "/" + language + ".docx", FileMode.Open, FileAccess.ReadWrite);

    localizationDict = ((item[])serializer.Deserialize(fileStream)).ToDictionary(i => i.id, i => i.value);

    fileStream.Close();
    isLoaded = true;

}
```

81. Funció FillDictionary Localització

La part final de la localització és buscar la clau especifica de cada text al diccionari i mostrar el text traduït correctament. Per fer això, tots els gameObjects amb el component Text, és a dir, els gameObjects que mostren algun text, tenen associat un script anomenat, TextLocalization. Aquest script és tan simple com especificar per cada Text la clau que tindrà al diccionari i fer una crida a la funció GetText de la classe LocalizationLogic, on buscarà la clau al text i li retornarà el valor.

```
public string key;

// Use this for initialization
void Start () {
    this.GetComponent<Text>().text = LocalizationLogic.GetText(key);
}
```

82.Classe TextLocalizatiom

Resultat Final

En aquest apartat és mostrarà el resultat final del joc, “Raid of Heroes”, per a PC, plataformes Windows, Linux i Mac.

Es detallarà e il·lustrarà el resultat i les accions que és poden dur a terme a cada una de les pantalles.

El videojoc comença amb una pantalla inicial, on apareix el títol del joc i es poden realitzar les següents accions:

- Canviar d’idioma el joc a través de les banderes situades a part superior dreta.
- Continuar jugant respecte la partida guardada.
- Iniciar una nova partida.
- Mostrar les Instruccions del joc.
- Sortir del joc.

83. Menú Principal

Les instruccions tenen el següent format, indicant cada tecla del teclat i ratolí a que estan associats al joc. Al fer clic a la creueta, et torna al menú principal.

84. Menú Instruccions

També des del menú principal es pot sortir del joc prement la tecla Esc. En prémer-la, apareix el menú de confirmació que hi ha a continuació on dient que sí, el videojoc es tanca. Si en canvi polses sobre el botó no, és tanca el popup i es queda al vista del menú principal.

85. Popup Tancar Joc

Solament es pot tancar l'aplicació, de manera segura, amb aquest procediment ja que és el lloc on ja s'ha guardat la informació i no hi ha perill de perdre cap progres.

Un cop l'usuari s'ha decidit a jugar, el primer cop que s'inicia el joc, no importa si inicia una nova partida o continua la partida, ja que no hi ha cap progres. Si l'usuari es passa algun nivell, tanca el joc i el torna a obrir, hauria de prémer sobre continuar la partida, altrament perdrà tot el progres que porti fins ara.

La pantalla que es mostra es la que hi ha a continuació. En el cas de la imatge, hi ha els dos primers nivells desbloquejats, però al iniciar una nova partida solament hi hauria el nivell 1.

Des d'aquest menú, l'usuari pot realitzar les següents accions:

- Tornar enrere, al menú principal.
- Seleccionar un nivell que estigui desbloquejat.

86. Menú Selecció Nivells

Quan l'usuari tria un nivell, el sistema carrega el nivell que ha premut i automàticament comença la partida. Les imatges que veiem corresponent al primer i segon nivell, d'esquerra a dreta. Com ja he comentat durant la memòria, hi ha diferents elements que interactuen amb el personatge, com els enemics, els Spikes, el cor i les claus.

Pel que fa a la UI, hi ha 3 objectes que aporten informació tant del jugador com de la pròpia partida i són els següents:

- *Vida del personatge*: La vida del personatge en format barra, el qual fa fàcil interpretar la vida restant. Baixa quan reps dany i puja quan agafes els cors.

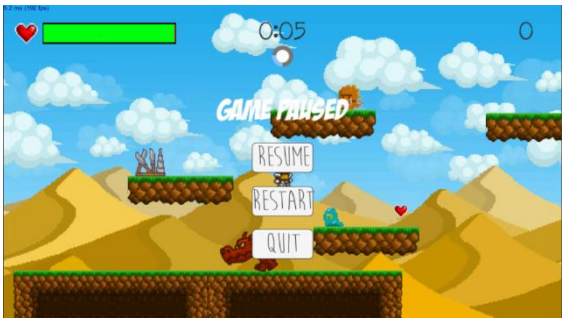
- *Temps transcorregut*: El temps que es porta de partida des del començament, en format minuts:segons. Les hores no estan contemplades ja que es molt poc factible que un usuari tardi aquest període de temps.
- *Puntuació*: Unitat que calcula la puntuació obtinguda durant la partida. Augmenta a cada atac culminat del personatge. Hi ha un bonus al acabar la partida depenent del temps que s'ha fet.

88. Inici Nivell 1

87. Inici Nivell 2

Des d'un nivell, l'usuari pot pausar el joc en tot moment, simplement prement la tecla ESC, apareix el menú de pausa de la següent imatge. En aquest punt, els enemics és congelen, el jugador no es pot moure i el temps, tant de les habilitats com el temps de partida, s'aturen. Des d'aquest menú es poden fer aquestes accions:

- Continuar Jugant: Aquesta acció descongela el joc i torna a fer funcionar el joc en el seu estat normal. Es pot arribar des de dos fronts diferents. El primer es prement la tecla Continuar o també, tornant a prémer la tecla ESC. Es tanca el menú de Pausa.
- Reiniciar Nivell: Tanca el menú de pausa i es torna a iniciar el nivell des de 0, resetejant vida, temps i puntuació.
- Tornar al Menú: Per últim, prement el botó sortir, tornarem al menú de selecció de nivells.



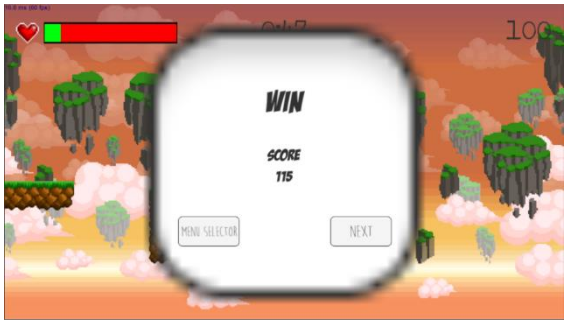
89. Menú Pausa durant el Joc

Un cop l'usuari arriba al final del nivell, és troba la porta i si ha trobat la clau, la porta s'obre per poder passar pel ella i acabar el nivell. Quan l'usuari està amb el personatge al davant de la porta, s'ha de prémer la tecla d'acció, la E, ja que sinó simplement fa l'acció de obrir-se la porta.

90. Porta per completar un nivell

Un cop finalitzat el nivell, apareix el popup de Victòria. En aquest popup és sumen els punts aconseguits al nivell, més el bonus del temps transcorregut. En aquest popup és poden realitzar dos accions:

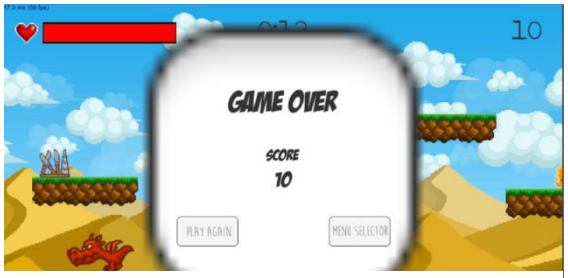
- *Menú de selecció de Nivell*: Et redirigeix al menú de selecció de nivell.
- *Següent*: És carrega el següent nivell, en ordre ascendent.



91. Popup Victòria

També existeix el popup de Derrota. Aquest popup pot aparèixer per vàries raons. La primera és si un enemic et mata i per tant la teva vida és zero. La segona és si el personatge és cau d'una plataforma. En ambdós casos el popup resultant es el següent. Com es pot observar, solament hi apareix la puntuació obtinguda durant la partida. Les accions que et deixa fer són les següents:

- *Tornar a Jugar*: Aquesta opció et deixa tornar a intentar el nivell, des de zero.
- *Menú de selecció de Nivell*: Et redirigeix al menú de selecció de nivell.



92. Popup Derrota

Testing

En aquest apartat parlaré una mica sobre el Testing que he realitzar per tal de comprovar, primer que els requeriments funcionals i no funcionals estan correctament implementats i segon que no hi hagi cap bug en les funcionalitats que he implementat. Per fer-ho he creat dos plantilles. La primera pels requeriments i la segona pels bugs.

Pel testing de bugs, he realitzat una build i els hi he passat a alguns amics, juntament amb la plantilla per tal que juguessin al joc i busquessin possibles bugs o possibles millores.

En el món del videojoc, els millors feedbacks i el millors testers són els usuaris finals ja que són qui jugaran al teu joc i per tant, els qui pagaran pel joc o d'alguna forma et generaran benefici.

Plantilles

Plantilla Requeriments

L'objectiu d'aquesta plantilla és comprovar que els requeriments estan ben implementats i compleix amb la funcionalitat requerida.

Identificador Test:	
Identificador Requeriment:	
Objectiu:	
Precondicions:	
Postcondicions:	
Compleix:	

- *Identificador Test*: Identificador de la Prova
- *IdRequeriment*: Identificador del Requeriment que estem comprovant.
- *Objectiu*: Funcionalitat a comprovar.
- *Precondicions*: Condicions necessàries per realitzar el test.
- *Postcondicions*: Condicions en que resultarà la prova.
- *Compleix*: Boolean verificant si aquell requeriment passa la prova.

Resultats

Identificador Test: T01	
Identificador Requeriment:	RF01
Objectiu:	Iniciar Nova Partida
Precondicions:	L'usuari es troba al menú principal i li dona a Iniciar Nova Partida
Postcondicions:	El joc es troba al menú de seleccionar un nivell. Solament hi ha el primer nivell desbloquejat
Compleix:	Sí

Identificador Test: T02	
Identificador Requeriment:	RF02
Objectiu:	Guardar partida
Precondicions:	L'usuari acaba una partida, ja sigui victòria o derrota.
Postcondicions:	El sistema ha generat un fitxer, a la ruta corresponent a persistentDataPath.
Compleix:	Sí

Identificador Test: T03	
Identificador Requeriment:	RF03
Objectiu:	Mostrar instruccions del joc.
Precondicions:	L'usuari es troba al menú principal i prem el botó Instruccions.
Postcondicions:	El sistema mostra un menú amb les instruccions
Compleix:	Sí

Identificador Test: T04	
Identificador Requeriment:	RF04
Objectiu:	Sortir de l'aplicació
Precondicions:	L'usuari es troba al menú principal i prem sobre la tecla Esc.
Postcondicions:	L'aplicació és tanca.
Compleix:	Sí

Identificador Test: T05	
Identificador Requeriment:	RF05
Objectiu:	Interacció amb el personatge
Precondicions:	L'usuari es troba a un nivell i prem una tecla que faci interactuar al personatge.
Postcondicions:	El personatge és mou en conseqüència a la tecla premuda.
Compleix:	Sí

Identificador Test: T06	
Identificador Requeriment:	RF06
Objectiu:	Reproduir música i/o efectes de so
Precondicions:	L'usuari es troba a un nivell i prem la tecla W o l'Espai.
Postcondicions:	Sona un efecte de so.
Compleix:	Sí

Identificador Test: T07	
Identificador Requeriment:	RF07
Objectiu:	Guardar estadístiques
Precondicions:	L'usuari acaba una partida, ja sigui victòria o derrota.
Postcondicions:	El sistema ha generat un fitxer, a la ruta corresponent a persistentDataPath.
Compleix:	Sí

Identificador Test: T08	
Identificador Requeriment:	RF08
Objectiu:	Seleccionar un nivell
Precondicions:	L'usuari està al menú de selecció de nivell.
Postcondicions:	El sistema entra al nivell que has seleccionat.
Compleix:	Sí

Identificador Test: T09	
Identificador Requeriment:	RF09
Objectiu:	Intel·ligència Artificial Enemics
Precondicions:	L'usuari es troba a un nivell amb diferents enemics.
Postcondicions:	El comportament dels enemics és diferent i existeix algun tipus d'intel·ligència artificial en ells.
Compleix:	Sí

Identificador Test: T10	
Identificador Requeriment:	RF10
Objectiu:	Menú Principal
Precondicions:	L'usuari obra el joc.
Postcondicions:	El sistema carrega el menú principal
Compleix:	Sí

Identificador Test: T11	
Identificador Requeriment:	RF11
Objectiu:	Menú Selecció Nivell
Precondicions:	L'usuari prem el botó Continuar Partida o Iniciar Nova Partida.
Postcondicions:	El sistema carrega el menú de selecció de nivell.
Compleix:	Sí

Identificador Test: T12	
Identificador Requeriment:	RNF01
Objectiu:	Compatibilitat Sistema Operatiu
Precondicions:	Executar el joc en diferents Sistemes Operatius.
Postcondicions:	El joc s'obre i funciona correctament.
Compleix:	Sí

Identificador Test: T13	
Identificador Requeriment:	RNF02
Objectiu:	Compatibilitat diferents Resolucions
Precondicions:	Executar el joc en diferents Resolucions
Postcondicions:	El joc és veu igual en totes elles
Compleix:	No

Després de veure el que passava, ho vaig mirar i sol vaig poder arreglar les resolucions on l'ample de la pantalla és més gran que l'alçada, en canvi, les més altes que llargues, les estretes, no les vaig poder arreglar i al final vaig decidir suportar solament resolucions amb el format 16:9 que és la resolució que fa servir gairebé la majoria de persones i així suporto diferents resolucions però dins d'un format 16:9.

També he deixat la opció de triar entre possibles resolucions o jugar amb mode finestra enlloc de obligar a jugar amb pantalla completa i a una sola resolució. Així obligo a que es faci servir un format en concret però dins de diferents resolucions,

Identificador Test: T14	
Identificador Requeriment:	RNF03
Objectiu:	Els texts han de ser llegibles.
Precondicions:	L'usuari obra l'aplicació i navega per ella.
Postcondicions:	L'usuari entén i pot llegir tots els texts
Compleix:	Sí

Identificador Test: T15	
Identificador Requeriment:	RNF04
Objectiu:	Localització
Precondicions:	L'usuari es troba al menú principal i prem el botó de Castellà.
Postcondicions:	El sistema canvia els texts a Castellà
Compleix:	Sí

Identificador Test: T16	
Identificador Requeriment:	RNF05
Objectiu:	Interfície Intuïtiva i Fàcil
Precondicions:	L'usuari entra a l'aplicació i navega per ella.
Postcondicions:	L'usuari entén tot el que fa i veu.
Compleix:	Sí

Identificador Test: T17	
Identificador Requeriment:	RNF06
Objectiu:	Fluïdesa
Precondicions:	L'usuari entra a un nivell.
Postcondicions:	El joc va a 60 fps estables i sense cap entrebanc.
Compleix:	Sí

Plantilla Bugs

L'objectiu d'aquesta plantilla és detectar els bugs que hi puguin haver i poder-los solucionar de la manera més eficient possible.

	Bug	Descripció	Passos/Accions per Reproduir-lo	Plataforma	Resolució	Es para el joc?	Trenca el joc?	Captures de pantalla				
1												
2												
3												
4												
5												

93. Plantilla Test Bugs

- *Bug*: Nom del bug o possible problema que hi pugui haver.
- *Descripció*: Text explicatiu i detallat del bug.
- *Passos/Accions per Reproduir-lo*: Els passos que ha seguit l'usuari perquè aquell bug aparegués.
- *Plataforma*: El tipus de plataforma en que ho estaven mirant, ja sigui Mac, Linux o Windows.
- *Resolució*: En quin aspecte estaven reproduint el joc.
- *Es para el joc?*: Booleana per saber si el joc, quan apareix el bug, és queda congelat.
- T*renca el joc*?: Booleana per saber si el joc, quan apareix el bug, és tanca l'aplicació.
- *Captures de pantalla*: Si pot ser, captures de pantalla per tenir una millor idea del que passa.

Resultats

En aquest apartat veurem els resultats obtinguts de la plantilla de bugs repartida entre uns quants amics. Per cada error que han reportat, intentaré donar una resposta a com ha estat tractat el bug com s'hauria de resoldre depenen del grau de dificultat o deelmolt que modificaria al joc. L'ordre dels resultats està segons la seva prioritat/importància, en ordre descendent, de més important a menys.

L'aspecte del report de bug serà amb vertical enlloc de horitzontal per millor llegibilitat.

Identificador Bug: B01	
Bug:	Drac Roig
Descripció:	El segon drac roig del nivell 1 me perseguia i quan s'ha acabat la plataforma seguia volant cap a l'esquerra
Passos per reproduir-lo:	1. Jugar al nivell 1 2. Ser perseguit per un Drac. 3. Al final de la plataforma, salta a una altra i fer que encara et persegueixi estant a la seva àrea de visió
Plataforma:	Windows

Aquest bug ja l'havia vist quan desenvolupava el joc, però no vaig saber trobar una manera de reproduir-lo, fins que hem va arribar aquest report i vaig entendre el que estava passant.

L'he pogut arreglar fent que simplement quan el drac arriba al límit d'una plataforma, doni la volta ignorant si esta o estava perseguint al personatge perquè vol dir que el personatge ja no està a la plataforma.

Aquest bug m'ha arribat per dos amics diferents i tot i que jo no l'hi donava importància al principi, veig i entenc que si que n'hi té.

Identificador Bug: B02	
Bug:	Imatge de Fons
Descripció:	Al caminar contra la vora d'una plataforma el fons avança tot i que el personatge no ho faci.
Passos per reproduir-lo:	1. Jugar a un nivell. 2. Quedar-se encallat contra una plataforma i donant-li a la tecla de moure's.
Plataforma:	Windows

He aconseguit millorar aquest problema tot i que no l'he pogut solucionar del tot. Ara per ara no és mou tant el fons quan estàs encallat a una paret, però per fer-ho bé i arreglar-ho necessitaria canviar el sistema de Colliders i també de la càmera i el personatge. Crec que seria massa feina pel poc que canviaria visualment.

Identificador Bug: B03	
Bug:	Vida Enemics
Descripció:	És molt incòmode no saber la vida que tenen els enemics
Passos per reproduir-lo:	1. Jugar a un nivell
Plataforma:	Windows

Aquest aspecte és una cosa que no havia tingut en compte. No he fet la solució al problema complert, però si que he provat a ficar una barra de vida sobre el cap dels enemics i activar-la quan reben dany i desactivar-la quan no. Crec que és una característica que millora molt la jugabilitat, ja que dona informació sobre els enemics i això aporta significativament al joc.

En aquesta imatge és pot veure el resultat, a petita escala, que he realitzat.

95. Barra de Vida Drac Roig 1

94. Barra de Vida Drac Roig 2

Identificador Bug: B04	
Bug:	Compte enrere propera habilitat
Descripció:	Estaria bé tindre un comptador per saber quan queda per activar l'habilitat
Passos per reproduir-lo:	1. Jugar a un nivell 2. Activar Habilitat de Personatge.
Plataforma:	Windows

És una característica que crec que restaria importància a les habilitats ja que si es sabes en quin moment està disponible, no es pararia de polsar i m'agrada pensar que l'habilitat és fa servir en moments de dificultat i per sortir del pas. Per tant és un bug que no crec que faci falta arreglar i és bo pel joc que estigui com està.

Identificador Bug: B05	
Bug:	Dificultat Enemics
Descripció:	Enemics massa difícils de matar. Més fàcil saltar-los i seguir.
Passos per reproduir-lo:	1. Jugar a un nivell
Plataforma:	Windows

És una de les característiques que més m'he plantejat durant tot el joc. Aquest report de bug és conseqüència de que els personatges i enemics tenen vida, és a dir, a jocs d'aquests tipus com Mario Bros, la vida dels enemics com la del personatge bé donada per vides. En el cas dels enemics tenen una vida, que en quan els toques és moren i normalment, igual passa amb el personatge.

En el meu joc, vaig voler donar-li aquesta vessant més de lluita , afegint més vida. Això fa que si el nivell és pot passar sense matar a tots els enemics, sigui més fàcil saltar-los.

Jo li veig tres possibles solucions, fer que el nivell és pugui passar, és a dir, s'obri la porta, quan tots els enemics estan morts, dona un sentit a matar als enemics. La segona possible solució és que la porta sol és pugui obrir a partir d'una puntuació mínima, que indirectament significa matar a X nombre d'enemics. I la tercera possible solució és canviar tot el sistema i fer que s'assembli més a jocs com Mario Bros i amb un sol toc es pugui matar als enemics. Això significaria treure tot el sistema de dany, armadura i vida com a número i fer que la vida dels enemics vaguin per vegades colpejats.

Crec que la millor opció seria obligar a que es matessin tots els enemics per tal de passar-se el nivell, ja que com he comentat, dona sentit a matar als enemics i apart, és la solució que menys canvis s'haurien de fer per dur a terme. Aquesta "millora" seria una característica a tenir en compte en un futur.

Identificador Bug: B06	
Bug:	Text Development Build
Descripció:	Surt "development console" allà al mig de tant en tant perquè sí.
Passos per reproduir-lo:	1. Obrir el joc
Plataforma:	Windows

Aquest bug, l'he ficat l'últim ja que realment no es un bug i no te gaire importància. Simplement apareixen les lletres Development Build" perquè és una versió de desenvolupament on poden haver bugs, errates o qualsevol altre problema. Per això a les versions primeres, com la alpha i la beta, normalment és fica el tet aquest.

Un cop és fa la build final, aquest text desapareix.

Conclusions

A continuació explicaré les conclusions personals que he tret d'aquest treball de fi de grau i les possibles línies futures que pot prendre l'aplicació per tal de millorar i seguir creixent.

Conclusions Personals

Primer de tot, la realització d'aquest treball m'ha fet recordar conceptes que he treballat durant el grau com tot el tema de l'anàlisi, els requeriments, casos d'ús i alguns conceptes que he utilitzat en la vessant de programació.

Gràcies a aquest treball he aconseguit incrementar i ampliar el meu coneixement tant de Unity, plataforma que no havia estudiat al grau però si havia fet anar durant les pràctiques tot i que no a un nivell com aquest, com de C#, un llenguatge que no havia aprés durant el grau però m'ha ajudat que sigui semblant al llenguatge Java que si havia programat amb ell. El principal objectiu que hem vaig plantejar al principi del treball de aprendre i comprendre com es realitza el desenvolupament d'un videojoc, crec que l'he complert, tot i que ha escala petita. Durant les pràctiques, vaig treballar amb videojoc que ja estaven o a la seva fase final desenvolupament o ja llençats al mercat i el que feia era arreglat bugs, per tant les tasques que he realitzat durant aquest treball són molt més complertes i complexes.

També cal destacar que he tingut algunes errates, tant de plantejament com durant el desenvolupament. Durant la preparació del treball, tenia la intenció de treballar amb la metodologia àgil, Scrum i em dol afirmar que no he treballat seguint aquesta metodologia, per diverses raons. Una de les raons es que estava jo sol desenvolupant el videojoc i es difícil aplica aquesta metodologia quan l'equip esta format solament per una persona. Jo havia de fer la figura de Product owner, d'equip de desenvolupament i de scrum master i es complicat tenir reunions amb mi mateix o ordenar el product backlog segons els diferents criteris, si solament hi ha el meu.

Durant el desenvolupament, hi ha coses que m'hagués agradat fer d'una altra manera, però per temps o per recursos no ha set possible. El tema del art és una d'aquetes coses que per falta de recursos, ja sigui un artista o comprar els assets d'internet, ha fluixejat. Els he tingut que recopilar d'aquí i d'allà, no tots tenen la mateixa estètica, no acaben d'estar ben polits i vulguis o no afecta al joc. Una altra característica que no he pogut implementar pel temps, ha set un sistema online de puntuacions per cada nivell. M'hagués agradat molt poder fer un sistema online per poder comparar les puntuacions dels diferents usuaris que han jugat als diferents nivells, ja que mai he tocat la vessant online de Unity, però no ha set possible ja que eren moltes hores i és complicat.

Tot i això, estic força orgullós i content del resultat final del videojoc, tot i les hores i hores que he passat programant i desenvolupant al Unity, ho he fet amb totes les ganes i sense perdre la il·lusió. Gràcies a les eines utilitzades com SourceTree, Asana, Visual Studio, m'han facilitat molt la feina i m'han permès agilitzar el procés de desenvolupament, ja que l'organització és un tema fonamental en el desenvolupament de qualsevol aplicació i/o videojoc.

Línies Futures

En un videojoc les línies futures que pot agafar sol molt diverses i quasi infinites ja que pot anar de arreglar bugs i ja, o canviar la dinàmica del joc de cap a peus., però intentaré expressar algunes idees pel desenvolupar noves característiques o millorar-ne per tal de fer anar creixent el joc. Les millores possibles que és poder dur a terme al videojoc són:

- Ampliar el nombre de plataformes, per poder ampliar el nombre de nivells, per tal que no es facin repetitius. Actualment hi ha dos tipus de plataformes i dos tipus de nivells, ampliar aquests nombres faria més atractiu i entretingut al joc.

- Incrementar el nombre de enemics. Estaria bé que hi haguessin dos o tres enemics per cada tipus de moviment/comportament que fan. Actualment hi ha 5 tipus de comportaments, els que et persegueixen, els que t'ataquen al veure't, els que simplement caminen d'un costat a l'altre, els que tenen esclaus i els que volen.

- Crear caps finals pel final d'alguns nivells. Aquests caps serien més poderosos que els simples enemics i suposarien un repte real.

- Com ja he comentat amb anterioritat, una de les opcions per millorar la jugabilitat i fer participar més activament als enemics, és modificar la manera de com és passa un nivell. Fer que la porta s'obri solament si tots els enemics han ser derrotats i no n'hi ha cap viu.

- També he comentat que hi podria haver un sistema online de puntuacions, per tal d'augmentar la competitivitat i obtenir la millor puntuació fos realment útil.

- Complementant amb la millora anterior, afegir que l'usuari pogués crear diferents comptes dins del joc, amb un nom d'usuari i un avatar.

- Una millora, potser una mica més remota, seria crear un editor de nivells i que la comunitat pugues desenvolupar els seus propis nivells i pujar-los online per que els usuaris puguin jugar.

Referències

1. http://unity3d.com/es/learn/tutorials Tutorials Unity
2. http://opengameart.org/ Open GameArt
3. http://www.gameart2d.com/ Game Art 2D
4. http://forum.unity3d.com/threads/loading-screen-between-scenes.204080/ Loading screen between scenes
5. http://forum.unity3d.com/threads/saving-and-loading-data-xmlserializer.85925/ Saving and Loading Data: XmlSerializer
6. https://www.youtube.com/watch?v=9bhkH7mtFNE Parallax Scrolling Background Unity
7. https://www.visualstudio.com/es-es/features/unitytools-vs.aspx VS Tools
8. https://unity3d.com/es Unity3D
9. https://www.sourcetreeapp.com/ Source Tree
10. https://app.asana.com/0/85688928674531/list Asana TFG Tasks
11. http://docs.unity3d.com/Manual/ Documentation Unity
12. http://www.gimp.org.es/ Gimp 2.0
13. http://stackoverflow.com/ StackOverflow
14. http://www.cocos2d.org/ Cocos2d
15. https://www.unrealengine.com/ UnrealEngine 4
16. https://bitbucket.org/fnicu6/tfg2016 Repositori TFG